Approximate Tree Decompositions of Planar Graphs in Linear Time

Frank Kammer and Torsten Tholey

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany
{kammer,tholey}@informatik.uni-augsburg.de

Abstract. Many algorithms have been developed for NP-hard problems on graphs with small treewidth k. For example, all problems that are expressable in linear extended monadic second order can be solved in linear time on graphs of bounded treewidth. It turns out that the bottleneck of many algorithms for NP-hard problems is the computation of a tree decomposition of width $O(k)$. In particular, by the bidimensional theory, there are many linear extended monadic second order problems that can be solved on n-vertex planar graphs with treewidth k in a time linear in n and subexponential in k if a tree decomposition of width $O(k)$ can be found in such a time.

We present the first algorithm that, on n-vertex planar graphs with treewidth k, finds a tree decomposition of width $O(k)$ in such a time. In more detail, our algorithm has a running time of $O(nk^3 \log k)$. The previous best algorithm with a running time subexponential in k was the algorithm of Gu and Tamaki [12] with a running time of $O(n^{1+\epsilon} \log n)$ and an approximation ratio $1.5 + 1/\epsilon$ for any $\epsilon > 0$. The running time of our algorithm is also better than the running time of $O(f(k) \cdot n \log n)$ of Reed's algorithm [17] for general graphs, where f is a function exponential in k.

Key words: tree decomposition, treewidth, branchwidth, rank-width, planar graph, ℓ-outerplanar, linear time, bidimensionality

1 Introduction

The treewidth, extensively studied by Robertson and Seymour [18], is one of the basic parameters in graph theory. Intuitively, the treewidth measures the similarity of a graph to a tree by means of a so-called tree decomposition. A tree decomposition of width r—defined precisely in Section 2—is a decomposition of a graph G into small subgraphs such that each subgraph contains only $r+1$ vertices and such that the subgraphs are connected by a tree-like structure. The treewidth $\mathrm{tw}(G)$ of a graph G is the smallest r for which G has a tree decomposition of width r.

The treewidth is often used to solve NP-hard problems on a graph G with small treewidth by the following two steps: First, compute a tree decomposition for G of a width r, and second, solve the problem using this tree decomposition. Unfortunately, there is a trade-off between the running times of the first and second step, depending on our choice of r. Very often the first step is the bottleneck. For example, Arnborg, Lagergren and Seese [3] showed that, for all problems expressible in so-called linear extended monadic second order (linear EMSO), the second step runs on n-vertex graphs in a time linear in n. Demaine, Fomin, Hajiaghayi, and Thilikos [8] have shown that, for many so-called bidimensional problems that are also linear EMSO problems, one can find a solution of a size ℓ in a given n-vertex graph—if one exists—in a time linear in n and subexponential in ℓ as follows: First, try to find a tree decomposition for G of width $r = \hat{c}\sqrt{\ell}$ for some constant $\hat{c} > 0$. One can choose $\hat{c}$ such that, if the algorithm fails, then there is no solution of size at most ℓ. Otherwise, in a second step, use the tree decomposition obtained to solve the problem by well known algorithms in $O(nc^r) = O(nc^{\hat{c}\sqrt{\ell}})$ time for some constant c. Thus, it is very important to support the first step also in such a time. This, even for planar graphs, was not possible by previously known algorithms.

In the following overview over related results, n denotes the number of vertices and k the treewidth of the graph under consideration. Tree decomposition and treewidth were introduced by Robertson and Seymour [18], which also presented the first algorithm for the computation of the treewidth and a tree decomposition polynomial in n and exponential in k [19]. There are numerous papers with improved running times, as, e.g., [2,7,15,21]. Here we focus on algorithms with running times either being polynomial in both, k and n, or being subquadratic in n. Bodlaender [5] has shown that a tree decomposition can be found in a time linear in n and exponential in k. However, the running time of Bodlaender's algorithm is practically infeasible already for very small k. The algorithm achieving the so far smallest approximation ratio of the treewidth among the algorithms with a running time polynomial in n and k is the algorithm of Feige, Hajiaghayi, and Lee [9]. It constructs a tree decomposition of width $O(k\sqrt{\log k})$ thereby improving the bound $O(k \log k)$ of Amir [1]. In particular, no algorithms with constant approximation ratios are known that are polynomial in the number of vertices and in the treewidth. One of the so far most efficient practical algorithms with constant approximation ratio was presented by Reed in 1992 [17]. His algorithm computes a tree decomposition of width $3k+2$ in $O(f(k) \cdot n \log n)$ time for some exponential function f. More precisely, this width is obtained after slight modifications as observed by Bodlaender [4].

Better algorithms are known for the special case of planar graphs. Seymour and Thomas [22] showed that the so-called branchwidth $\mathrm{bw}(G)$ and a so-called branch decomposition of width $\mathrm{bw}(G)$ for a planar graph G can be computed in $O(n^2)$ and $O(n^4)$ time, respectively. A minimum branch decomposition of a graph G can be used directly—like a tree decomposition—to support efficient algorithms. For each graph G, its branchwidth $\mathrm{bw}(G)$ is closely related to its treewidth $\mathrm{tw}(G)$; in detail, $\mathrm{bw}(G) \leq \mathrm{tw}(G) + 1 \leq \max(3/2\,\mathrm{bw}(G), 2)$ [20]. Gu and Tamaki [11] improved the running time to $O(n^3)$ for constructing a branch decomposition and thus for finding a tree decomposition

of width $O(\mathrm{tw}(G))$. Recently they also showed that one can compute a tree decomposition of width $(1.5+c)\mathrm{tw}(G)$ for a planar graph G in $O(n^{(c+1)/c}\log n)$ time for each $c \geq 1$. It is unknown whether a tree decomposition of smallest width for planar graphs can be found in polynomial time.

For planar graphs with n vertices and treewidth $k \geq 1$, our algorithm computes a tree decomposition of width $O(k)$ in $O(n \cdot k^3 \log k)$ time. This means that we obtain a better running time than Gu and Tamaki for all planar graphs of treewidth $k \leq n^{o(1)}$. Graphs with a larger treewidth are usually out of interest since for such graphs it is not clear whether we can efficiently solve the second step of our two steps defined for solving NP-hard problems.

Our result can be used to find a solution of size $\ell \geq 1$ for many bidimensional graph problems that are expressible in linear EMSO in a time linear in n and subexponential in ℓ. Such problems are, e.g., Minimum Dominating Set, Minimum Maximal Matching, and Minimum Vertex Cover, which all are NP-hard on planar graphs.

In contrast to general graphs, on planar graphs many graph parameters as branchwidth and rank-width differ only by a constant factor from the treewidth [10,16,20]. For that reason, our algorithm can be also used to find a constant approximation for these parameters on planar graphs in a time linear in n.

2 Main Ideas

In this section, we sketch the ideas of our algorithm. For an easier intuition, we want to compare embedded planar graphs (G, φ) with landscapes by assigning a *height* to each vertex v referred to as $h_\varphi(v)$. The set of all vertices incident to the outer face is called the *coast* (of the embedded graph), and we define the height of each vertex of the coast to be 1. We then define inductively the set of vertices of *height* $i > 1$ as the set of vertices that are part of the coast of the subgraph obtained by removing all vertices of height smaller than i. A graph is called *ℓ-outerplanar* if there is an embedding φ such that all vertices have height at most ℓ. In this case, φ is also called *ℓ-outerplanar*. For a subgraph G' of an embedded graph (G, φ), we use $\varphi|_{G'}$ to denote the embedding of G restricted to the vertices and edges of G'. For a graph $G = (V, E)$ and a vertex set $V' \subseteq V$, we let $G[V']$ be the subgraph of G induced by the vertices of V'; and we define $G - V'$ to be the graph $G[V \setminus V']$. If a graph G is a subgraph of another graph G', we write $G \subseteq G'$. For a graph $G = (V, E)$, we also say that a vertex set $S \subseteq V$ *disconnects* two vertex sets $A, B \subseteq V$ *weakly* if no connected component of $G - S$ contains vertices of both A and B. S *disconnects* A and B *strongly* if additionally $S \cap (A \cup B) = \emptyset$ holds. If a vertex set S strongly disconnects two non-empty vertex sets, we say that S is a *separator* (for these vertex sets).

Definition 1 (tree decomposition, bag, width). *A* tree decomposition *for a graph $G = (V, E)$ is a pair (T, B), where $T = (V_T, E_T)$ is a tree and B is a function that maps each node w of T to a subset of V—called the* bag *of w—such that*

1. *each vertex of G is contained in a bag and each edge of G is a subset of a bag,*
2. *for each vertex $v \in V$, the nodes whose bags contain v induce a subtree of T.*

The width *of (T, B) is $\max_{w \in V_T}\{|B(w)| - 1\}$.*

As observed by Bodlaender [6], one can easily construct a tree decomposition of width $3k-1$ for a k-outerplanar graph $G = (V, E)$ in $O(k|V|)$ time. One idea to find a tree decomposition of width $O(k)$ for an $\omega(k)$-outerplanar graph of treewidth k is to search for a separator Y of size $O(k)$ that

disconnects the vertices of large height strongly from the coast by applying the following theorem. We also call a set Y that strongly disconnects a vertex set U from the coast a *coast separator* (for U).

Theorem 2. *Let (G, φ) be an embedded graph of treewidth $k > 1$. Moreover, let V_1 and V_2 be connected sets of vertices of G such that $(\min_{v \in V_2} h_\varphi(v)) - (\max_{v \in V_1} h_\varphi(v)) \geq k + 1$. Then, there exists a set Y of size at most k that strongly disconnects V_1 and V_2.*

Proof. We first consider the case that V_1 and V_2 consist of only one vertex v_1 and v_2, respectively. Let (T, B) be a tree decomposition of width k with a smallest number of bags containing both, v_1 as well as v_2. If there is no such bag, then for the two closest nodes w_1 and w_2 in T with $v_i \in B(w_i)$ $(i \in \{1, 2\})$ and for the node w' adjacent to w_2 on the w_1-w_2-connecting path in T, the set $B(w') \cap B(w_2)$ is a separator of size at most k and it strongly disconnects V_1 and V_2. Hence, let us assume that there is at least one node w in T with its bag containing v_1 and v_2. Since $|B(w)| \leq k + 1$, for at least one number i with $h_\varphi(v_1) < i < h_\varphi(v_2)$, there is no vertex of height i in $B(w)$. Since $h_\varphi(v_2) > i$, there is a connected set Y that consists exclusively of vertices of height i with Y disconnecting $\{v_2\}$ from all vertices u with $h_\varphi(u) < i$, i.e., in particular from $\{v_1\}$. The nodes of T, whose bags contain at least one vertex of Y, induce a subtree T' of T by the properties of a tree decomposition. Let us root T such that w is a child of the root and such that the subtree T_w of T rooted in w does not contain any node of T'. We then replace T_w by two copies T_1 and T_2 of T_w and similarly the edge between the root r of T and the root of T_w by two edges connecting r with the root of T_1 and T_2, respectively. In addition, we define the bag of each node w' in T_1 to consist of those vertices of the bag $B(w')$ of (T, B) that are also contained in the connected component of $G[V \setminus Y]$ containing v_2. The bag of each node w' in T_2 should contain the remaining vertices of $B(w')$. The replacement described above leads to a tree decomposition of width k with a lower number of bags containing v_1 as well as v_2. Contradiction.

If V_1 or V_2 consists of more than one vertex, we define G_2 to be the subgraph of G induced by all vertices of G with their height being at least $\min_{v \in V_2} h_\varphi(v)$. Let C be the connected component of G_2 containing V_2. Let us consider the graph obtained from G by merging the vertices of height at most $\max_{v \in V_1} h_\varphi(v)$ to one vertex v_1 and by merging the vertices of C to one vertex v_C. Additionally add two new vertices v' and v'' and three new edges $\{v_1, v'\}, \{v', v''\}$, and $\{v'', v_1\}$ into the graph. For the graph G' obtained, one can obviously find an embedding φ' such that the coast of B' consists of the vertices v_1, v', and v'' and such that $h_{\varphi'}(v_C) - h_{\varphi'}(v_1) = \min_{v \in V_2} h_\varphi(v) - \max_{v \in V_1} h_\varphi(v)$. Thus, any separator of size at most k disconnecting v_C and v_1 in G' is also a separator in G with the properties stated in the theorem. □

Let us define a maximal connected set H of vertices of the same height to be a *crest* if no vertex of H is connected to a vertex of larger height. For a planar embedded graph G with treewidth $k \in \mathbb{N}$ and one crest, we can try to construct a tree decomposition with the following algorithm: Initialize $G'_1 = (V'_1, E'_1)$ with G. For $i = 1, 2, \ldots,$ as long as G'_i has only one crest and has vertices of height at least $2k + 1$, apply Theorem 2 to obtain a coast separator Y_i separating the vertices of height at least $2k + 1$ from all vertices of height at most k. Then, define $G'_{i+1} = (V'_{i+1}, E'_{i+1})$ as the subgraph of G'_i induced by the vertices of Y_i and of the connected component of $G'_i \setminus Y_i$ that contains the crest of G'_i. Moreover, let $G_i = G'_i[Y_i \cup (V'_i \setminus V'_{i+1})]$. If the recursion stops with a $(2k + 1)$-outerplanar graph G'_j $(j \in \mathbb{N})$, we set $G_j = G'_j$ and construct a tree decomposition for G as follows: First, determine a tree decomposition (T_i, B_i) of width $O(k)$ for each G_i $(i \in \{1, \ldots, j\})$

This is possible since G_i is $O(k)$-outerplanar. Second, set $Y_0 = Y_j = \emptyset$. Then, for all $i \in \{1, \dots, j\}$, add the vertices of $Y_i \cup Y_{i-1}$ to all bags of (T_i, B_i). Finally, for all $i \in \{1, \dots, j-1\}$, connect an arbitrary node of T_i with an arbitrary node of T_{i+1}. This leads to a tree decomposition for G.

If we are given a planar embedded graph G with treewidth $k \in \mathbb{N}$ that has more than one crest of height at least $2k+1$ (or if this is true for one of the subgraphs G'_i defined above) we cannot apply Theorem 2 to find one coast separator separating simultaneously all vertices of height at least $2k+1$ from the coast since these vertices may not be connected. For cutting off the vertices of large height, one might use several coast separators—one for each connected component induced by the vertices of height at least $2k+1$. However, if we insert the vertices of all coast separators into a tree decomposition of the remaining graph consisting of the vertices with a low height, this may increase the width of the tree decomposition too much since there may be more than a constant number of coast separators. This is the reason why we search for further separators called *perfect crest separators* that are disjoint from the crests and that partition our graph G (or G'_i) into smaller subgraphs—called *components*—such that, for each component C containing a non-empty set V' of vertices of height at least $2k+1$, there is a set Y_C of vertices with the following properties:

(P1) Y_C is a coast separator of size $O(k)$ for V'.
(P2) Y_C is contained in C.
(P3) Each vertex of Y_C has height at least $k+1$.

The main idea is that—in some kind similar to the construction above—we want to construct a tree decomposition separately for each component and afterwards to combine these tree decompositions to a tree decomposition of the whole graph. The properties above should guarantee that, for each tree decomposition computed for a component, we have to add the vertices of at most one coast separator into the bags of that tree decomposition. We next try to guarantee (P1)-(P3).

By making the components so small that each component has at most one maximal connected set of vertices of height at least $2k+1$, we can easily find a set of coast separators satisfying (P1) and (P3) (Theorem 2). We next try to find some constraints for the perfect crest separators such that we can also guarantee (P2). Let us assume that our graph is triangulated. This simplifies the proofs and the definitions. For example, in a triangulated graph the vertices of a coast separator induce a unique cycle. Suppose for a moment that it is possible to choose each perfect crest separator as the vertices of a path with the property that, for each pair of consecutive vertices u and v, the height of v is one smaller than the height of u. In Section 3, we call such a path a *down path* if it also has some additional nice properties. For a down path P, there is no path that connects two vertices u and v of P being strictly shorter than the subpath of P from u to v. Consequently, whenever we search for a coast separator of small size for a connected set V' in a component C, there is no need to consider any coast separator with a subpath Q consisting of vertices that are strongly disconnected from the vertices of C by the vertex set of P. Note that it is still possible that vertices of a perfect crest separator belong to a coast separator. To guarantee that (P2) holds, in a more precise definition of the components, we let each edge e of a crest separator belong to two components. Then we can observe that, if we choose the perfect crest separators as down paths not containing any vertex of any crest, each maximal connected set of vertices of height at least $2k+1$ in a component C has a coast separator Y_C that is completely contained in C.

Unfortunately, one down path can not be a separator. Therefore, the idea is to define a perfect crest separator as the vertices of the union of two down paths P_1 and P_2 that start in the same vertex or in two adjacent vertices. For more information on crest separators, see Section 3. We also write $P_1 \circ P_2$ to denote the concatenation of the reverse path of P_1, a path Q, and the path P_2,

where Q is the path induced by the edge connecting the first vertex of P_1 and the first vertex of P_2, or the empty path, if P_1 and P_2 start in the same vertex. Unfortunately, with our new definition of a perfect crest separator we cannot avoid in general that a coast separator crosses a perfect crest separator. Thus (P2) may be violated. To omit this problem we define a *minimal coast separator* for a connected set S in a graph G to be a coast separator Y for S that consists of a minimal number of vertices such that among all such coast separators the subgraph of G induced by the vertices of Y and the vertices of the connected component of $G \setminus Y$ containing S has a minimal number of inner faces. In Section 4 (Lemma 19), we observe that, if a minimal coast separator Y_C for a connected set of vertices of height at least $2k+1$ in a component C passes through another component C', Y_C also separates all vertices of height at least $2k+1$ in C' from the coast.

For an intuition, let us define an (s-t-)*ridge* as a path from a vertex s to a vertex t such that, among all paths from s to t, the minimum height of a vertex on the ridge is as large as possible. This height is called the *depth* of the ridge. Let us now consider a ridge R connecting two vertices of height $2k+1$ that belongs to two different connected components H_1 and H_2 in the subgraph of our original graph induced by the vertices of height at least $2k+1$. Note that R has a depth of at most $2k$. Let us assume for simplicity that a vertex v of lowest height on R has neighbors v_1 and v_2 on 'both sides' of R such that their height is one smaller than the height of v—for the remaining cases, see Lemma 6. Then, we can find a perfect crest separator X that strongly disconnects the vertices of H_1 and H_2 and that consists of two down paths P_1 and P_2 with the first edge of P_i ($i \in \{1,2\}$) being $\{v, v_i\}$. For simplicity, we assume in the following that P_1 and P_2 are vertex-disjoint apart from their common first vertex. Note that X consists of at most $4k-1$ vertices since v has height at most $2k$. Let C_i ($i = 1, 2$) be the connected component of $G-X$ that contains H_i. If a cycle induced by a minimal coast separator Y for H_1 in G also contains vertices of C_2, then this cycle contains a subpath P with its endpoints being different vertices x_1 and x_2 of $P_1 \circ P_2$. The minimality of Y implies that P has a shorter length than the x_1-x_2-connecting subpath P' of $P_1 \circ P_2$. Because P is shorter than P', P cannot contain any vertex of a height larger than that of v. We can also conclude that x_1 and x_2 do not both belong to the same path P_i ($i \in \{1,2\}$). Consequently, the cycle induced by Y must also enclose H_2. In this case, we merge the components containing H_1 and H_2 to one super component C^* so that afterwards both properties (P1) and (P2) hold for C^*. For guaranteeing property (P3) instead of a real minimal coast separator, we take one that is minimal among all coast separators without any vertex of height lower than $k+1$. More details are described in Section 5.

Given a perfect crest separator X consisting of the vertices of two paths P_1 and P_2, the idea is to find tree decompositions (T_1, B_1) and (T_2, B_2) for the two components of G on either sides of $P_1 \circ P_2$ such that T_i, for each $i \in \{1,2\}$, has a node w_i with $B_i(w_i)$ containing all vertices of P_1 and P_2. By inserting an additional edge $\{w_1, w_2\}$ we then obtain a tree decomposition for the whole graph. However, in general we are given a set $\mathcal{X}$ of perfect crest separators that splits our graph into components for which (P1)-(P3) holds. If we want to construct a tree decomposition for a component, we possibly have to guarantee that for more than one perfect crest separator $X \in \mathcal{X}$ there is a bag containing all vertices of X. Since we can use the techniques described above to cut off the vertices of very large height from each component, it remains to find such a tree decomposition for the remaining $O(k)$-outerplanar subgraph of the component. Because of the simple structure of our perfect crest separators we can indeed find such a tree decomposition by a slightly modified version of an algorithm of Bodlaender [6] for $O(k)$-outerplanar graphs. For more details see Section 6. Finally, we can iteratively connect the tree decompositions constructed for

the several components in the same way as it is described in case of one perfect crest separator. For an example, see also Fig. 1 and 2.

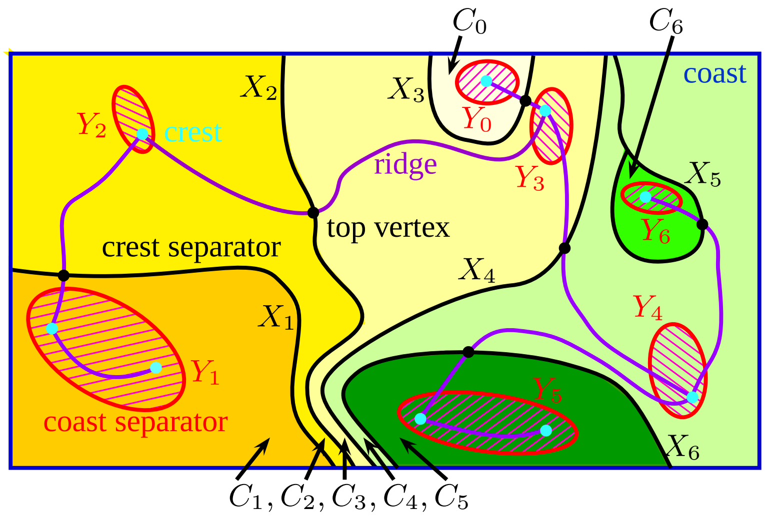

Fig. 1. A planar embedded graph (G, φ) decomposed by a set $\mathcal{S} = \{X_1, \ldots, X_6\}$ of crest separators into several components $C_0, \ldots, C_6$. In addition, each component C_i has a coast separator Y_i for all crests in C_i.

3 Decomposition into mountains

We already defined a crest as a maximal connected set H of vertices of the same height in an embedded planar graph if no vertex in H is connected to a vertex of larger height. An embedded planar graph with exactly one crest is called a *mountain*. In this section we want to show in detail how we can split an embedded planar graph (G, φ) into several mountains. Our splitting process is much simpler on *almost triangulated graphs* that are embedded planar graphs in which the boundary of each inner face consists of exactly three vertices and edges. If, for a graph G of treewidth k with a planar embedding φ, (G, φ) is not almost triangulated, it can be replaced by an almost triangulated graph (G', φ') with $\mathrm{tw}(G') \leq 4k + 1$ by simply adding into each inner face a new vertex and by connecting this vertex with all vertices on the boundary of that inner face (see Theorem 2 in [14]). Another simplification is to consider only biconnected graphs, which we also call *bigraphs*. For a non-biconnected graph, one can easily construct a tree decomposition for the whole graph by combining tree decompositions for each biconnected component. Thus, we describe our splitting process for an almost triangulated bigraph. The good news is that our splitting process will split almost triangulated bigraphs in almost triangulated bigraphs, whereas triangulated or triconnected graphs could be splitted into graphs that are not triangulated or triconnected anymore. However, we want to mention that one can obtain a better approximation ratio of the treewidth by translating our techniques to non-triangulated graphs. This is much more complicated and can be found in [13].

As indicated in Section 2 our splitting process makes use of so-called perfect crest separators and down paths. Unfortunately, it is not so easy to compute perfect crest separators. Therefore

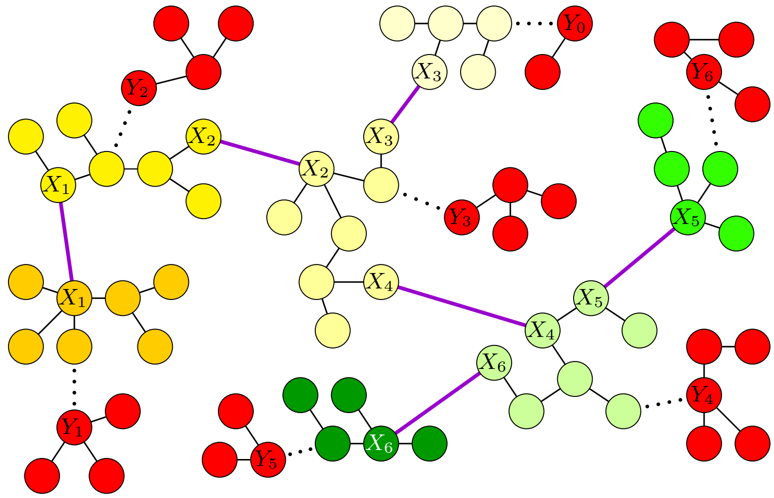

Fig. 2. A sketch of a tree decomposition for the graph G of Fig 1. In detail, each colored component C_i has its own tree decomposition (T_i, B_i) for the vertices of small height in C_i with the bags of (T_i, B_i) being colored with the same color than the component C_i. We implicitly assume that Y_i is part of all bags of (T_i, B_i). A tree decomposition for the vertices of large height in C_i is constructed recursively and contains a bag—marked with Y_i—containing all vertices Y_i and being connected to one bag of (T_i, B_i). For connecting the tree decompositions of different components, there is an edge connecting a node w' of (T_i, B_i) with a node w'' of (T_j, B_j) if and only if C_i and C_j share a certain edge (on their boundary) that later is defined more precisely. In this case, the bags w' and w'' contain the vertices of the crest separators disconnecting C_i and C_j—and are marked by the name of this crest separator.

our next goal is to define a set of so-called crest separators, which is a superset of perfect crest separators. First of all, we have to define down path precisely. Let us assume w.l.o.g. that the vertices of all graphs considered in this paper are numbered with pairwise different integers. For each vertex $u \in V$ with a height $q \geq 2$, we partition its neighbors into sets of maximal size such that the vertices of each set appear around u immediately one after each other and have either all height $q-1$ or all height at least q. For each such set S, if it consists of vertices of height $q-1$, we define the vertex $v \in S$ with the smallest integer to be a *neighborhood representant for each vertex in S around u*. Moreover, for each vertex u with a height $q \geq 2$, we define the *down vertex* of u to be the neighbor of u that among all neighbors of u with height $q-1$ has the smallest number. We denote the down vertex of u by $u\!\downarrow$. The down edge of u is the edge $\{u, u\!\downarrow\}$. The *down path* (of a vertex v) is a path (that starts in v), that consists completely of down edges, and that ends in a vertex of the coast. Note that in an almost triangulated graph every vertex has a down path since in an almost triangulated graph each vertex of height at least 2 is connected to a vertex of a smaller height and a vertex v of height 1 is a down path of zero length.

Definition 3 (crest separator). *A* crest separator *in an almost triangulated graph is a tuple $X = (P_1, P_2)$ with P_1 being a down path starting in some vertex u and P_2 either*

- *being a down path starting in a neighbor of u with the same height than u or*

– *being a path that, for a neighborhood representant $v \neq u{\downarrow}$ around u, starts with the edge $\{u, v\}$ and ends with the down path of v.*[1]

The first vertex $u' \neq u$ on P_1 that also is part of P_2, if it exists, is called the lowpoint *of X.*

In the remainder of this paper, for an almost triangulated graph (G, φ), we let $\mathcal{S}(G, \varphi)$ be the set of all crest separators in (G, φ). Note that, for a crest separator $X = (P_1, P_2)$, the vertex set of $P_1 \circ P_2$ usually defines a separator. This explains the name crest separator, but formally a crest separator is a tuple of paths. Since an n-vertex planar graph has at most $O(n)$ edges, the next lemma follows from Definition 3.

Lemma 4. *Given an almost triangulated ℓ-outerplanar graph (G, φ) with n vertices, the set $\mathcal{S}(G, \varphi)$ can be constructed in $O(\ell n)$ time.*

Since crest separators are in the main focus of our paper, it is useful to introduce some additional terminology: The *top vertices* of a crest separator $X = (P_1, P_2)$ consist of the first vertex of P_1 and the first vertex of P_2—possibly this is only one vertex. For a crest separator $X = (P_1, P_2)$, we write $v \in X$ and say that v is a vertex of X to denote the fact that v is a vertex of P_1 or P_2. The *border edges* of X are the edges of $P_1 \circ P_2$. The *top edge* of X is the only border edge that is not a down edge. Note that this means that the top edge either connects the first vertex of P_1 with the first vertex of P_2 or is the first edge of P_2. The *height* of X is the maximum height over all its vertices. The *essential boundary of* X is the subgraph of G induced by the border edges of X that appear on exactly one of the two paths P_1 and P_2. In particular, if X has a lowpoint, the vertices of the essential boundary consists exactly of the vertices appearing before the lowpoint on P_1 or P_2 and of the lowpoint itself. For two vertices s_1 and s_2 being part of the essential boundary of X, the *crest path* from s_1 to s_2 is the shortest path from s_1 to s_2 that consists only of border edges of X and that does not contain the lowpoint of X as an inner vertex. If s_1 and s_2 are vertices of $P_1 \circ P_2$, but not both are part of the essential boundary of X, the *crest path* from s_1 to s_2 is the shortest path from s_1 to s_2 consisting completely of border edges.

For an almost triangulated graph (G, φ), each set $\mathcal{S}$ of crest separators splits G into several subgraphs. More precisely, for a set $\mathcal{S} \subseteq \mathcal{S}(G, \varphi)$, let us define two inner faces F and F' of (G, φ) to be $(\mathcal{S}, \varphi)$-connected if there is a list $(F_1, \ldots, F_\ell)$ ($\ell \in \mathbb{N}$) of inner faces of (G, φ) with $F_1 = F$ and $F_\ell = F'$ such that, for each $i \in \{1, \ldots, \ell - 1\}$, the faces F_i and F_{i+1} share a common boundary edge not being a border edge of a crest separator in $\mathcal{S}$. A set $\mathcal{F}$ of inner faces of (G, φ) is $(\mathcal{S}, \varphi)$-connected if each pair of faces in $\mathcal{F}$ is $(\mathcal{S}, \varphi)$-connected. Hence, a graph is splitted by crest separators into the following kind of subgraphs.

Definition 5 ($(\mathcal{S}, \varphi)$-component). *Let $\mathcal{S} \subseteq \mathcal{S}(G, \varphi)$ for an almost triangulated graph (G, φ). For a maximal nonempty $(\mathcal{S}, \varphi)$-connected set $\mathcal{F}$ of inner faces of (G, φ), the subgraph of G consisting of the set of vertices and edges that are part of the boundary of at least one face $F \in \mathcal{F}$ is called an* $(\mathcal{S}, \varphi)$-component.

For a single crest separator X in an almost triangulated graph (G, φ), the set $\{X\}$ splits (G, φ) into exactly two $(\{X\}, \varphi)$-components, which, for an easier notation, are also called (X, φ)-*components*. For the two (X, φ)-components $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$, we say that X *goes weakly between* two vertex sets U_1 and U_2 if $U_1 \subseteq V_1$ and $U_2 \subseteq V_2$ or if $U_1 \subseteq V_2$ and $U_2 \subseteq V_1$. If

[1] Note that the existence of a neighborhood representant implies that u in this case has height ≥ 2.

additionally $U_1 \cup U_2$ does not contain any vertex of X, we say that X *goes strongly between* the sets.[2]

We need to find a subset of the set of all crest separators to separate all crests from each other. Recall that, for vertices s and t, an *s-t-ridge* is a path connecting s and t such that, among all paths with these endpoints, the minimum height of a vertex on the ridge is as large as possible, and this minimum height is called the *depth* of the ridge. A *mountain structure* is a tuple $(G, \varphi, \mathcal{S})$ with (G, φ) being an almost triangulated bigraph and $\mathcal{S} \subseteq \mathcal{S}(G, \varphi)$ such that, for each pair of different crests H_1 and H_2 in (G, φ) and for each ridge R in (G, φ) with one endpoint in H_1 and the other in H_2, the following property holds:

(a) There is a crest separator $X \in \mathcal{S}$ that has a height equal to the depth of R, that strongly goes between H_1 and H_2, and that among all such crest separators in $\mathcal{S}(G, \varphi)$ has as few as possible top vertices.

The next lemma shows that the simple structure of a crest separator as the combination of two down paths suffices to separate the crests of the graph; in other words, property (a) can be easily satisfied for $\mathcal{S} = \mathcal{S}(G, \varphi)$.

Lemma 6. *Let s and t be vertices, and let R be an s-t-ridge in an almost triangulated bigraph (G, φ) such that R, among all s-t-ridges, has the smallest number of vertices with their height being equal to the depth of R. Take v as a vertex of R with smallest height. If the depth of R is smaller than the height of both s and t, there is a crest separator $X \in \mathcal{S}(G, \varphi)$ that goes strongly between $\{s\}$ and $\{t\}$, that contains v as a top vertex, and that contains no border edge being part of R.*

Proof. Take $G = (V, E)$. Let us say that a vertex $u \in V \setminus \{s, t\}$ on R is *dominated on the right* (*left*) of R if all edges not on R leaving u to the right (left) side of R lead to a vertex with height $h_\varphi(u){+}1$ and if u on the right (left) side of R is not incident to the outer face. Note that this latter condition is superfluous in the case $h_\varphi(u) \geq 2$. In the following, let i be the height of v.

The first case that we consider is the case, where $i \geq 2$ and v is dominated neither on the left nor on the right of R. In particular, v is connected by an edge to a vertex $u \in V \setminus \{s, t\}$ of height $i - 1$ or i with the edge $\{v, u\}$ leaving R to a different side than the edge $\{v, v{\downarrow}\}$. If $h_\varphi(u) = i - 1$, there is a crest separator $X \in \mathcal{S}(G, \varphi)$ using $\{v, u'\}$ as top edge, where u' is the neighborhood representant of u around v. Note that X goes strongly between $\{s\}$ and $\{t\}$ since $v{\downarrow}$ and u' are on different sides of R and since R is a path not visiting any vertex of height $i - 1$, i.e., R cannot be crossed by a down path from $v{\downarrow}$ or from u'. If $h_\varphi(u) = i$, there is a crest separator $X \in \mathcal{S}(G, \varphi)$ using $\{v, u\}$ as top edge. In this case, u is connected to the vertex $u{\downarrow}$ of height $i - 1$ and $u{\downarrow}$ is different from $v{\downarrow}$ since u and $v{\downarrow}$ leave R on different sides. Again, X goes strongly between $\{s\}$ and $\{t\}$. Note that the border edges of the crest separators constructed above does not contain any edge of R.

In the second case, we assume that $i = 1$ and v is dominated neither on the left nor on the right of R. In this case v must be incident to the outer face on at least one side of R. If v also on the other side of R is incident to the outer face, then $\{v\}$ is a separator of size 1 which is impossible since G is biconnected. Otherwise, v dominated neither on the left nor on the right implies that

[2] These definitions focus on the disconnection of faces instead of vertex sets. In particular, if a crest separator X weakly (strongly) goes between to non-empty vertex sets A and B, then the set of vertices of X weakly (strongly) disconnects A and B. Note also that a usual separator disconnects at least two non-empty vertex sets whereas this must not be the case for the vertex set of a crest separator.

there is a vertex u also having height 1 such that $\{u, v\}$ is not incident to the outer face. Then the down paths of u and v, which are of zero length, define a crest separator with the desired properties.

Finally, we consider the remaining case in which v is dominated on the left or right side of R. Since G is almost triangulated, we then can replace v by a list L of its neighbors all having a larger height than v and obtain an s-t-path R' either having a larger depth than R or having the same depth as R, but a smaller number of vertices with a height equal to the depth of R. In both cases, this contradicts our choice of R. Note explicitly that this is true even if L contains s or t. $\square$

Thus, for each almost triangulated bigraph (G, φ), the tuple $(G, \varphi, \mathcal{S}(G, \varphi))$ is a mountain structure for (G, φ). It appears that some crest separators of a mountain structure may be useless since they split one crest into several crests or they cut of parts of our original graph not containing any crests. This means that it is often useful to restrict the set of crest separators of a mountain structure. Hence, we define a mountain structure $(G, \varphi, \mathcal{S})$ to be *good* if also the following properties hold:

(b) No crest separator in $\mathcal{S}$ contains a vertex of a crest in (G, φ).
(c) Each $(\mathcal{S}, \varphi)$-component contains vertices of a crest in (G, φ).

Note that the properties (a) and (c) imply that each $(\mathcal{S}, \varphi)$-component contains the vertices of exactly one crest in (G, φ). We next want to show that a good mountain structure exists and can be computed efficiently. For that let us define a *mountain connection tree* T of a mountain structure $(G, \varphi, \mathcal{S})$ to be a graph defined as follows: Each node of T is identified with an $(\mathcal{S}, \varphi)$-component of G, and two nodes w_1 and w_2 of T are connected if and only if they—or more precisely the $(\mathcal{S}, \varphi)$-components with which they are identified—have a common top edge.

Recall that the border edges of a crest separator $X = (P_1, P_2)$ consist of one top edge e of X and further down edges, and that a down edge cannot be a top edge of any crest separator. Hence, the top-edge e is the only top edge of a crest separator that is contained in both (X, φ)-components. This means that, for a crest separator $X \in \mathcal{S}$ of a good mountain structure, we can partition the set of all $(\mathcal{S}, \varphi)$-components into a set $\mathcal{C}_1$ of $(\mathcal{S}, \varphi)$-components completely contained in one (X, φ)-component and the set $\mathcal{C}_2$ of $(\mathcal{S}, \varphi)$-components contained in the other (X, φ)-component. Then, X is the only crest separator with a top edge belonging to $(\mathcal{S}, \varphi)$-components in $\mathcal{C}_1$ as well as in $\mathcal{C}_2$. Consequently, T is indeed a tree.

Lemma 7. *The mountain connection tree of a mountain structure is a tree.*

The fact that the top edge of a crest separator X is the only top edge belonging to both (X, φ)-components shows also the correctness of the following lemma.

Lemma 8. *Let C_1 and C_2 be two $(\mathcal{S}, \varphi)$-components that are neighbored in the mountain connection tree of a mountain structure $(G, \varphi, \mathcal{S})$. Then there is exactly one crest separator going weakly between C_1 and C_2, which is also the only crest separator with a top edge belonging to both C_1 and C_2.*

Using a simple breadth-first search on the dual graph of G one can easily determine the $(\mathcal{S}, \varphi)$-components of a mountain structure and prove the following lemma.

Lemma 9. *Given a set $\mathcal{S}$ of crest separators of a mountain structure $(G, \varphi, \mathcal{S})$, the mountain connection tree can be easily determined in $O(|V|)$ time, where V is the vertex set of G.*

We also can construct a good mountain structure.

Lemma 10. *Given an almost triangulated ℓ-outerplanar bigraph (G, φ) with n vertices, one can construct a good mountain structure $(G, \varphi, \mathcal{S})$ in $O(\ell n)$ time.*

Proof. For a simpler notation in this proof we call a crest separator of a set $\tilde{\mathcal{S}}$ of crest separators to be a *largest crest separator* of $\tilde{\mathcal{S}}$ if it has a largest height among all crest separators in $\tilde{\mathcal{S}}$ and if, among all crest separators of largest height, it has a maximal number of top vertices.

We first construct the set $\mathcal{S}' = \mathcal{S}(G, \varphi)$ of all crest separators in $O(\ell n)$ time (Lemma 4). Lemma 6 guarantees that, for each pair of crests, there is a crest separator in $\mathcal{S}(G, \varphi)$ strongly going between the two crests and that it can be chosen in such a way that one of its top vertices is a vertex of lowest height on a ridge between the two crests. This in particular means that no vertex of the crest separator is part of a crest in G. Hence, in $O(\ell n)$ time, we can remove all crest separators from $\mathcal{S}'$ that contain a vertex of a crest. This does not violate property (a). Afterwards property (b) of a good mountain structure holds. Let $\mathcal{S}''$ be the resulting set of crest separators. For guaranteeing property (c), we have to remove further crest separators from $\mathcal{S}''$. Therefore, we construct the mountain connection tree T of $(G, \varphi, \mathcal{S}'')$ in $O(n)$ time (Lemma 9).

In a sophisticated bottom-up traversal of T we then remove superfluous crest separators in $O(\ell n)$ time. For a better understanding, before we present a detailed description of the algorithm, we roughly sketch some ideas. Our algorithms marks some nodes a finished in such a way that the following invariant always holds: If a node C of T is marked as finished, the $(\mathcal{S}'', \varphi)$-component C contains a crest of (G, φ). The idea of the algorithm is to process a so far unfinished node that has only children already marked as finished and that, among all such nodes, has the largest depth in T. When processing a node w, we possibly remove a crest separator X from the current set $\mathcal{S}''$ of crest separators with the top edge of X belonging to the two $(\mathcal{S}'', \varphi)$-components identified with w and a neighbor w' of w in T. If so, by the replacement of $\mathcal{S}''$ by $\mathcal{S}'' \setminus \{X\}$, we merge the nodes w and w' in T to a new node w^*. We also mark w^* as finished only if w' is a child of w since in this case we already know that w' is already marked as finished and w' and hence also w^* therefore contains the vertices of a crest in (G, φ).

We now come to a detailed description of our algorithm starting with some preprocessing steps. In $O(n)$ time, we determine and store with each node w of T a value $\mathrm{Crest}(w) \in \{0, 1\}$ that is set to 1 if and only if the $(\mathcal{S}'', \varphi)$-component identified with w contains a vertex that is part of a crest of (G, φ). Within the same time, we mark additionally each node as unfinished and store with each non-leaf w of T in a variable $\mathrm{MaxCrestSep}(w)$ a largest crest separator of the set of all crest separators going weakly between the $(\mathcal{S}'', \varphi)$-component identified with w and an $(\mathcal{S}'', \varphi)$-component identified with a child of w. For each leaf w of T, we define $\mathrm{MaxCrestSep}(w) = \mathrm{nil}$. As a last step of our preprocessing phase, which also runs in $O(n)$ time, for each node w of T, we initialize a value $\mathrm{MaxCrestSep}^*(w)$ with nil. $\mathrm{MaxCrestSep}^*(w)$ is defined analogously to $\mathrm{MaxCrestSep}(w)$ if we restrict the crest separators to be considered only to those crest separators that go weakly between two $(\mathcal{S}'', \varphi)$-components identified with w and with a finished child of w. We will see that it suffices to know the correct values of $\mathrm{MaxCrestSep}(w)$ and $\mathrm{MaxCrestSep}^*(w)$ only for the unfinished nodes and therefore we do not update these values for finished nodes. We next describe the processing of a node w during the traversal of T in detail. Keep in mind that $\mathcal{S}''$ is always equal to the current set of remaining crest separators, which is updated dynamically.

First we exclude the case, where w has a parent $\tilde{w}$ with an unfinished child $\hat{w}$, and where $\mathrm{MaxCrestSep}(\tilde{w})$ is equal to the crest separator going weakly between the $(\mathcal{S}'', \varphi)$-components

identified with w and $\tilde{w}$. More precisely, in this case we delay the processing of w and continue with the processing of $\hat{w}$. This is possible since $\hat{w}$ has the same depth than w.

Second, we test whether the $(\mathcal{S}'', \varphi)$-component C identified with w contains a vertex belonging to a crest of (G, φ), which is exactly the case if $\mathrm{Crest}(w) = 1$. In this case, we mark w as finished. If there is a parent $\tilde{w}$ of w with X being the crest separator going weakly between the two $(\mathcal{S}'', \varphi)$-components identified with w and to $\tilde{w}$, we replace $\mathrm{MaxCrestSep}^*(\tilde{w})$ by a largest crest separator in $\{X, \mathrm{MaxCrestSep}^*(\tilde{w})\}$.

Let us next consider the case where $\mathrm{Crest}(w) = 0$. We then remove a largest crest separator X of the set of all crest separators going weakly between the $(\mathcal{S}'', \varphi)$-component identified with w and an $(\mathcal{S}'', \varphi)$-component identified with a neighbor of w. X must be either $\mathrm{MaxCrestSep}(w)$ or the crest separator going weakly between the two $(\mathcal{S}'', \varphi)$-components identified with w and to its parent. If X is a crest separator going weakly between the two $(\mathcal{S}'', \varphi)$-components identified with w and a child $\hat{w}$ of w, we remove X from $\mathcal{S}''$ and mark the node w^* obtained from merging w and $\hat{w}$ as finished and set $\mathrm{Crest}(w^*) = 1$. Note that this is correct since $\hat{w}$ is already marked as finished and therefore $\mathrm{Crest}(\hat{w}) = 1$. In addition, if there is a parent $\tilde{w}$ of w, we replace $\mathrm{MaxCrestSep}^*(\tilde{w})$ by the largest crest separator contained in $\{X', \mathrm{MaxCrestSep}^*(\tilde{w})\}$ for the crest separator $X' \in \mathcal{S}''$ going strictly between the $(\mathcal{S}'', \varphi)$-components identified with w and $\tilde{w}$.

Otherwise, X is the crest separator going weakly between the two $(\mathcal{S}'', \varphi)$-components identified with w and to its parent $\tilde{w}$. In this case, we mark the node w^* obtained from merging the unfinished nodes w and $\tilde{w}$ as unfinished, set $\mathrm{Crest}(w^*) = \mathrm{Crest}(\tilde{w})$, and define the value $\mathrm{MaxCrestSep}^*(w^*)$ as the largest crest separator in $\{\mathrm{MaxCrestSep}^*(w), \mathrm{MaxCrestSep}^*(\tilde{w})\}$ or nil if this set contains no crest separator. If $\tilde{w}$ beside w has another unfinished child, we define $\mathrm{MaxCrestSep}(w^*)$ as the largest crest separator in $\{\mathrm{MaxCrestSep}(w), \mathrm{MaxCrestSep}(\tilde{w})\}$. Here we use the fact that $\mathrm{MaxCrestSep}(\tilde{w}) \neq X$ since otherwise the processing of w would have been delayed. If $\tilde{w}$ has no other unfinished child, we take $\mathrm{MaxCrestSep}(w^*)$ as the largest crest separator that is contained in $\{\mathrm{MaxCrestSep}(w), \mathrm{MaxCrestSep}^*(\tilde{w})\}$ or nil if no crest separator is in this set. Note that $\mathrm{MaxCrestSep}^*(\tilde{w}) \neq X$ since w is unfinished before its processing.

By induction one can easily show that the values $\mathrm{MaxCrestSep}(w)$ and $\mathrm{MaxCrestSep}^*(w)$ for all unfinished nodes w and $\mathrm{Crest}(w)$ for all nodes w are updated correctly. If the processing of a node in T is delayed, the processing of the next node considered is not delayed. Hence the running time is dominated by the non-delayed processing steps. If the processing of a node is not delayed either two $(\mathcal{S}'', \varphi)$-components are merged or a node w in T is marked as finished. Hence the algorithm stops after $O(n)$ processing steps, i.e., in $O(n)$ time, with all nodes of T being marked as finished. Since it is easy to see that the algorithm marks a node as finished only if its identified $(\mathcal{S}'', \varphi)$-component contains the vertices of a crest in (G, φ) and that it never merges two $(\mathcal{S}'', \varphi)$-components with both components containing vertices of a crest in (G, φ), the invariant holds before and after each processing of a node w. Thus, $(G, \varphi, \mathcal{S}'')$ defines a good mountain structure at the end of the algorithm. $\square$

A good mountain structure $(G, \varphi, \mathcal{S})$ splits our graph in that sense into mountains that every $(\mathcal{S}, \varphi)$-component contains exactly one crest of (G, φ). This is good enough for most of our applications of good mountain structures. However, given an $(\mathcal{S}, \varphi)$-component C of a good mountain structure $(G, \varphi, \mathcal{S})$ and the embedding φ' of C obtained by restricting φ to the vertices and edges of C, the graph (C, φ') needs not to be a mountain since the height of the vertices in G and C may differ with respect to their corresponding embeddings. Therefore in the next section we extend

the $(\mathcal{S},\varphi)$-components to so-called extended components, which are mountains. The property that extended components are mountains is then used in Section 6.

4 Connection between coast separators and pseudo shortcuts

By definition, a good mountain structure $(G,\varphi,\mathcal{S})$ contains exactly one crest of (G,φ) in each $(\mathcal{S},\varphi)$-component. As part of our algorithm, for each $(\mathcal{S},\varphi)$-component, we want to compute a coast separator that strongly disconnects its crest of (G,φ) from the coast and that is of size $c\cdot\mathrm{tw}(G)$ for some constant c. Unfortunately, for some $(\mathcal{S},\varphi)$-components C, there might be no coast separator with the desired properties that is completely contained in C. However, for every connected set M of vertices in an almost triangulated graph, for which there is a coast separator whose vertex set induces a unique cycle. If such a cycle is not completely contained in one $(\mathcal{S},\varphi)$-component C, there is a crest separator $X\in\mathcal{S}$ such that the cycle contains a subpath P starting and ending in vertices of X in the (X,φ)-component not containing C. Then, either P can be replaced by a crest path without increasing the length or P must be a pseudo shortcut.

Definition 11 ((h-high s_1-s_2-connecting) (D-)pseudo shortcut). *Let $X=(P_1,P_2)$ be a crest separator in an almost triangulated graph (G,φ), and let D be an (X,φ)-component. Let s_1 and s_2 be both vertices of the essential boundary of X.*[3] *Then, a path from s_1 to s_2 in D is called an* (s_1-s_2-connecting) (D-)pseudo shortcut (of X) *if it has a strictly shorter length than the crest path from s_1 to s_2 and if the path does not contain any vertex of the coast. The pseudo shortcut is called* h-high *if all its vertices have height at least h.*

The reason for restricting the endpoints of a pseudo shortcut P to be part of the essential boundary is that, otherwise, one of P_1 and P_2 must contain both endpoints. Then P cannot have a shorter length than the subpath of P_1 or P_2 connecting the two endpoints as shown by part (a) of the next lemma.

Lemma 12. *Let $X=(P_1,P_2)$ be a crest separator.*

(a) *For each $i\in\{1,2\}$, no path with its endpoints v_1 and v_2 in P_i can have a shorter length than the v_1-v_2-connecting crest path.*
(b) *If X is of height ℓ and has $i\in\{1,2\}$ top vertices, each pseudo shortcut of X can visit at most $i-1$ vertices of height ℓ and no vertex of height larger than ℓ.*

Proof. Both (a) and (b) hold since the heights of adjacent vertices differ by at most 1. ☐

Intuitively speaking, the endpoints of a pseudo shortcut for a crest separator X in an almost triangulated graph (G,φ) are the vertices between which a coast separator can leave one (X,φ)-component and later reenter the (X,φ)-component. For an efficient computation of coast separators, we therefore want to compute the vertex pairs between which a pseudo shortcut of X exists. For that we need some further definitions and technical lemmata. For a mountain structure $(G,\varphi,\mathcal{S})$ and the (X,φ)-components D and $\tilde{D}$ of a crest separator $X\in\mathcal{S}$, we say that $\tilde{D}$ is *opposite* to D. Additionally, we call a subpath P' of a path P in G to be a *maximal (D,X)-subpath* of P if it is a path in D that visits at least one vertex not in X and if no edge of D appears immediately before or after P' on P.

[3] Note that s_1 and s_2 are vertices of D.

For a cycle Q in an embedded graph (G, φ) not containing any vertex of the coast, we say that Q *encloses* a vertex u, a vertex set U, and a subgraph H of G, if the set of the vertices of the cycle is a coast separator for u, U, and the vertex set of H, respectively. The subgraph of G induced by the vertices of Q and the vertices enclosed by Q is the *inner graph* of Q. We also say that a crest separator X with a lowpoint *encloses* a vertex u, a vertex set U, and a subgraph H of G, if u, U and the vertex set of H, respectively, are enclosed by the cycle induced by the edges of the essential boundary of X. For a mountain structure $(G, \varphi, \mathcal{S})$, an h-high s_1-s_2-connecting D-pseudo shortcut P of a crest separator $X \in \mathcal{S}$ is called *strict* if it has shortest length among all h-high s_1-s_2-connecting D-pseudo shortcuts and if among those the inner graph of the cycle induced by the pseudo shortcut and the crest path from s_2 to s_1 of X has a minimal number of inner faces.

Lemma 13. *Let $(G, \varphi, \mathcal{S})$ be a good mountain structure, let D be an (X, φ)-component for a crest separator $X \in \mathcal{S}$, and let P be a strict h-high s_1-s_2-connecting D-pseudo shortcut of X. Then, for all crest separators $X' \in \mathcal{S}$ with an (X', φ)-component $D' \subseteq D$, P has at most one maximal (D', X')-subpath. If P has such a subpath, this path is a strict h-high D'-pseudo shortcut of X'.*

Proof. Let $X' = (P'_1, P'_2)$ be a crest separator in $\mathcal{S}'$ with an (X', φ)-component $D' \subseteq D$. Assume first that P contains two or more maximal (D', X')-subpaths. See Fig. 3 for an illustration. Thus, for some $i \in \{1, 2\}$, P contains two vertices s' and s'' on P'_i such that the crest path P' of X' connecting s' and s'' does not belong completely to P, but is part of the inner graph of the cycle Q consisting of P and the crest path of X connecting the endpoints of P. Hence, we can replace the subpath of P connecting s' and s'' by P', which also replaces Q by a new cycle Q'. This replacement does not increase the length of P (Lemma 12.a). However, since P' does not completely follow P but is part of the inner graph of Q, the inner graph of Q' has a smaller number of inner faces than Q. Hence P cannot be a strict pseudo shortcut. Contradiction. If P contains exactly one maximal (D', X') path P', then, because of P being a strict h-high D-pseudo shortcut, P' must be a strict D'-pseudo shortcut of X'. $\square$

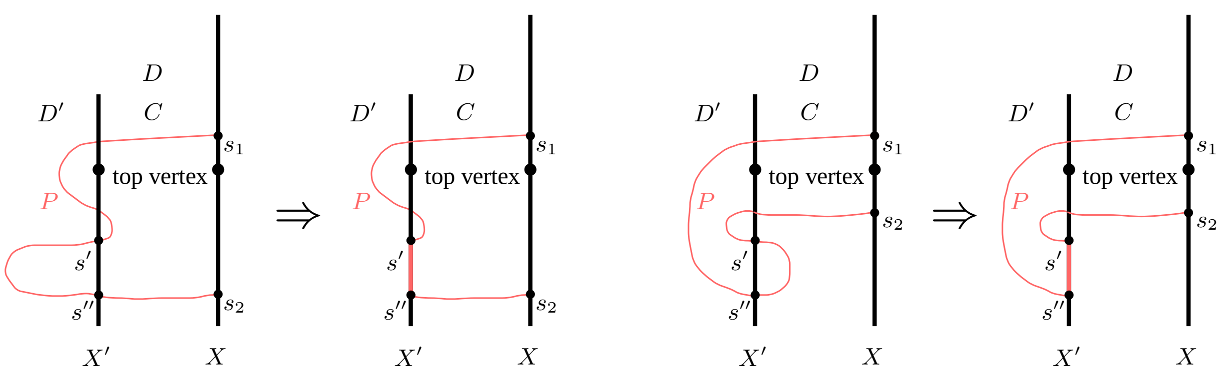

Fig. 3. The replacement of a pseudo shortcut P crossing X' more than once.

In the following we want to compute pseudo shortcuts for the different crest separators by a bottom-up traversal in the mountain connection tree T of a good mountain structure $(G, \varphi, \mathcal{S})$. We hope that, for a crest separator X going weakly between an $(\mathcal{S}, \varphi)$-component C and its father

in T and for the (X, φ)-component D containing C, a D-pseudo shortcut can be constructed as follows: It first follows a path in C and possibly, after reaching a vertex v of C that is part of a crest separator $X' \in \mathcal{S}$ weakly going between C and a child C' of C in T, it follows a (precomputed) D'-pseudo shortcut for the (X', φ)-component D' containing C', then it follows again a path in C and, after possibly visiting further pseudo shortcuts, it returns to C and never leaves C anymore. Indeed, if a D-pseudo shortcut, immediately after reaching a vertex v of C that is part of a crest separator X' with the properties described above, visits a vertex outside C, then we show that it contains a D'-pseudo shortcut P' for an (X', φ)-component D' with the properties described above. However some endpoints of P' possibly do not belong to C since X' may not completely contained in C. Similarly, since X also may not completely contained in C, a D-pseudo shortcut may not start in a vertex of C. This is the reason why, for subsets $\mathcal{S}', \mathcal{S}'' \subseteq \mathcal{S}(G, \varphi)$ of an almost triangulated graph (G, φ) with $\mathcal{S}' \subseteq \mathcal{S}''$ and for an $(\mathcal{S}', \varphi)$-component C, we define the *extended component* $\mathrm{ext}(C, \mathcal{S}'')$ as the graph obtained from C by adding the border edges of all crest separators $X \in \mathcal{S}''$ with a top edge in C. This should mean of course that the endpoints of the border edges are also added as vertices to C.

The next lemmata prove some properties of extended components that lead to an important relation between extended components and pseudo shortcuts described in Lemma 17. This relation helps to construct pseudo shortcuts efficiently as shown in the proof of Lemma 18.

Lemma 14. *Let (G, φ) be an almost triangulated graph, let C be an $(\mathcal{S}, \varphi)$-component for a set $\mathcal{S} \subseteq \mathcal{S}(G, \varphi)$, and let e be an edge with exactly one endpoint v in $\mathrm{ext}(C, \mathcal{S})$. Then, v is part of a crest separator in $\mathcal{S}$ with a top edge in C.*

Proof. By the definition of $\mathrm{ext}(C, \mathcal{S})$ the assertion of the lemma holds if v is not contained in C. It remains to consider the case that v is in C. The definition of an $(\mathcal{S}, \varphi)$-component implies that $v \in X$ for some crest separator $X \in \mathcal{S}$. Let u be the vertex of largest height contained in C such that the down path P of u contains v and such that the subpath from u to v of P is contained in C. If there is a top edge $\{u, u'\}$ for some vertex u' part of C, then v is part of a crest separator with a top edge in C. Otherwise, u is on the boundary of C, i.e., u is incident to two boundary edges e_1 and e_2 of C. As boundary edges they must be either down edges or top edges. No endpoint of e_1 and e_2 is higher than u since, otherwise, we have chosen u in the wrong way. Thus, only one edge of $\{e_1, e_2\}$ can be a down edge and the other is a top edge. This is a contradiction since we are in the case with no top edge $\{u, u'\}$ being part of C. □

For an (X, φ)-component D of a crest separator X in an almost triangulated graph (G, φ) and for any set $\mathcal{S}$ with $\{X\} \subseteq \mathcal{S} \subseteq \mathcal{S}(G, \varphi)$, we next show two auxiliary lemmata.

Lemma 15. *The extended component $\mathrm{ext}(D, \mathcal{S})$ consists only of the vertices and edges of D, the border edges of X, and their endpoints.*

Proof. Each vertex $v \in \mathrm{ext}(D, \mathcal{S})$ either belongs to D or is reachable by a down path from a vertex of D. A down path P starting in a vertex of D can leave D only after reaching a vertex $x \in X$. But after reaching a vertex $x \in X$, P must follow the down path of x and therefore all edges after x on P must be border edges of X. □

Lemma 16. *If e is an edge with exactly one endpoint v in $\mathrm{ext}(D, \mathcal{S})$, then v is part of X.*

Proof. Let $\tilde{D}$ be the (X, φ)-component opposite to D. Then, there are no direct edges from a vertex $v \in D$ to a vertex $\tilde{v} \in \tilde{D}$ with neither v nor $\tilde{v}$ being part of X. Hence, v must be part of X by Lemma 15. □

For the rest of this section, let us assume that we are given a fixed good mountain structure $(G, \varphi, \mathcal{S})$ with $G = (V, E)$. Let T be the mountain connection tree of $(G, \varphi, \mathcal{S})$ with an arbitrary node of T chosen as root.

Lemma 17. *Let $C, C_0, C_1, \ldots, C_j$ be $(\mathcal{S}, \varphi)$-components such that, in T, the father of C is C_0 and the children of C are $C_1, \ldots, C_j$. Moreover, for $i \in \{0, 1, \ldots, j\}$, let X_i be the crest separator with a top edge in C and C_i. Then, for the (X_0, φ)-component D that contains C, a strict h-high D-pseudo shortcut of X_0 can only consist of subpaths in $\mathrm{ext}(C, \mathcal{S})$ and of strict h-high D_i-pseudo shortcuts of X_i for the (X_i, φ)-components D_i that contain C_i ($i \in \{1, \ldots, j\}$).*

Proof. Clearly, D consists exclusively of the vertices and edges in C and in $D_1, \ldots, D_j$. Let us consider a strict h-high D-pseudo shortcut P of X_0 that is not completely contained in $\mathrm{ext}(C, \mathcal{S})$. Take e to be the first edge of P not contained in $\mathrm{ext}(C, \mathcal{S})$. Hence, one endpoint u of e must be part of a crest separator X_i with $i \in \{1, \ldots, j\}$—here we use Lemma 14 and the fact that $X_0, X_1, \ldots, X_j$ are the only crest separators with a top edge in C—and the other endpoint $v \notin \mathrm{ext}(C, \mathcal{S})$ is contained in D_i. For the (X_i, φ)-component $\tilde{D}_i$ opposite to D_i, this means that v is not contained in $\mathrm{ext}(\tilde{D}_i, \mathcal{S})$. However, there must be a next vertex contained in $\mathrm{ext}(\tilde{D}_i, \mathcal{S})$. Because of Lemma 16 the next vertex v' of $\mathrm{ext}(\tilde{D}_i, \mathcal{S})$ after v on P must be also part of X_i. Since P is a strict h-high D-pseudo shortcut of X_0, by Lemma 13 it has at most one maximal D_i-subpath, and the subpath of P from u to v' must be a strict h-high D_i-pseudo shortcut of X_i. If there are further parts of P not contained in $\mathrm{ext}(C, \mathcal{S})$, they can similarly shown to be strict h-pseudo shortcuts for one of the other crest separators in $\{X_1, \ldots, X_j\} \setminus \{X_i\}$. □

For an (X, φ)-component D of a crest separator $X \in \mathcal{S}$, let us define an *h-high D-pseudo shortcut set for X* to be a set consisting of a strict s_1-s_2-connecting h-high D-pseudo shortcut for each pair $\{s_1, s_2\}$ of vertices of X for which such a pseudo shortcut exists. By applying the last lemma several times, one can compute pseudo shortcut sets for all crest separators.

Lemma 18. *Assume that all crest separators in $\mathcal{S}$ have height at most q. Let $h \geq 2$ be an integer. Then, in $O(|V|q^3)$ total time, one can compute an h-high D-pseudo shortcut set for all crest separators $X \in \mathcal{S}$ and all (X, φ)-components D.*

Proof. We already know that we can compute the mountain connection tree T of $(G, \varphi, \mathcal{S})$ in $O(|V|)$ time (Lemma 9). We then compute the pseudo shortcut sets in a bottom-up traversal of T followed by a top-down traversal. Intuitively, the bottom-up traversal computes the pseudo shortcut sets for the (X, φ)-components below crest separators X whereas the top-down traversal computes the pseudo shortcut sets for the (X, φ)-components above crest separators X. More precisely, let $C, C_0, C_1, \ldots, C_j$ be $(\mathcal{S}, \varphi)$-components, let $X_0, \ldots, X_j$ be crest separators, and let $D, D_1, \ldots, D_j$ be (X_0, φ)-$, \ldots, (X_j, \varphi)$-components chosen such as in Lemma 17. Then in a bottom-up traversal, we can assume that we have already computed an h-high D_i-pseudo shortcut set $\mathcal{L}(D_i)$ for X_i for all $i \in \{1, \ldots, j\}$. By Lemma 17 we can compute an h-high D-pseudo shortcut set $\mathcal{L}(D)$ for X_0 by just starting a single-source shortest path algorithm for each vertex s of X_0 as source vertex on the following graph C'. The graph C' is obtained from $\mathrm{ext}(C, \mathcal{S})$ by deleting all vertices v with

$h_\varphi(v) \leq h-1$ and by inserting an edge for each pseudo shortcut in $\mathcal{L}(D_i)$ such that the edge connects the endpoints of the pseudo shortcut. Moreover, for each edge e introduced for a pseudo shortcut, we assign the length of the pseudo shortcut as weight to e, whereas we assign weight 1 to each edge of C. Before analyzing the running time, let us define $m_{C'}$ to be the number of edges in C', and n_C to be the number of vertices in C. Since all pseudo shortcuts have a length of at most $2q$, we can terminate the single-source shortest path computation after the computation of the distance from s for all vertices for which this distance is smaller than $2q$. Therefore and since there are at most q^2 pseudo shortcuts for one crest separator in $\mathcal{S}$, each of the $O(q)$ single source-shortest paths problems on C' can be solved in $O(m_{C'}+q) = O(n_C + (j+1)q^2)$ time. Note that $j = \deg_T(w) - 1$ for the node w in T identified with C and that the number of nodes in T is $O(|V|)$. Therefore the whole running time for the bottom-up traversal—including the computation of the graph C' for each extended component $\mathrm{ext}(C,S)$—is $O(|V|q^3)$ time. Afterwards in a top-down traversal, for each crest separator X, we can compute a D-pseudo shortcut set for the (X,φ)-component D that was not already considered. Clearly this takes the same time than the bottom-up traversal. $\square$

In Section 5 we construct a set of pairwise non-crossing coast separators of three different types. In the remaining part of this section two of those types are introduced. We also study under which conditions a coast separator can cross another coast separator or can cross a crest separator since this is useful to guarantee that the coast separators constructed in Section 5 do not cross.

For introducing the first type of coast separators let us call a pseudo shortcut to be *ℓ-long* if it consists of at most ℓ edges. For a crest separator X and an (X,φ)-component D, let us define an *ℓ-long h-high D-pseudo shortcut set* for X to be a set obtained from an h-high D-pseudo shortcut set for X by removing all pseudo shortcuts of length greater than ℓ. For a pseudo shortcut P of a crest separator X leading from a vertex $s \in X$ to a vertex $t \in X$, we call the cycle consisting of P and the crest path of X from t to s to be the *composed cycle* of (X,P). The first type of coast separators that we consider are composed cycles of (X,P) for a crest separator X and an ℓ-long h-high pseudo shortcut P. We next study under which conditions pseudo shortcuts can cross.

Let us consider a fixed path $\tilde{P}$ in the mountain connection tree T of $(G,\varphi,\mathcal{S})$ as shown in Fig. 4 . To get an easier intuition, let us assume that the vertices of $\tilde{P}$ are ordered from left to right. For some $r \in I\!N$, take $C_0,\ldots,C_r$ as the $(\mathcal{S},\varphi)$-components being the nodes of $\tilde{P}$ from left to right. Define X_i $(1 \leq i \leq r)$ to be the crest separator whose top edge belongs to both C_{i-1} and C_i. For $i = 0,\ldots,r$, let C_i^+ be the $(\{X_1,\ldots,X_r\},\varphi)$-component containing C_i. Let H_i be the crest of (G,φ) contained in C_i $(i \in \{0,\ldots,r\})$. For a crest separator X_i $(i \in \{1,\ldots,r\})$ and the (X_i,φ)-component D that contains C_i, we call the D-pseudo shortcuts the *right pseudo shortcuts of* X_i, and all $(\mathcal{S},\varphi)$-components contained in D are the $(\mathcal{S},\varphi)$-components *right from* X_i. Similarly, the $\tilde{D}$-pseudo shortcuts for the (X,φ)-component $\tilde{D}$ opposite to D are the *left pseudo shortcuts of* X_i, and all $(\mathcal{S},\varphi)$-components contained in $\tilde{D}$ are the $(\mathcal{S},\varphi)$-components *left from* X_i.

Lemma 19. *For each left* (right) *pseudo shortcut P of a crest separator X_i with $i \in \{1,\ldots,r\}$, the composed cycle of (X_i,P) encloses the crest contained in C_{i-1}* (in C_i).

Proof. Let P' be the composed cycle of (X_i,P), and let R be a ridge between H_{i-1} and H_i. If P' does not enclose H_{i-1}, (does not enclose H_i), P must visit a vertex u of R. From the fact that $(G,\varphi,\mathcal{S})$ is a good mountain structure it follows that the height of the top vertices of X_i is equal to the depth of R. By Lemma 12.b we then can conclude that X_i must have two top vertices, that u must have exactly the same height as the top vertices and that u is the only vertex on P of this

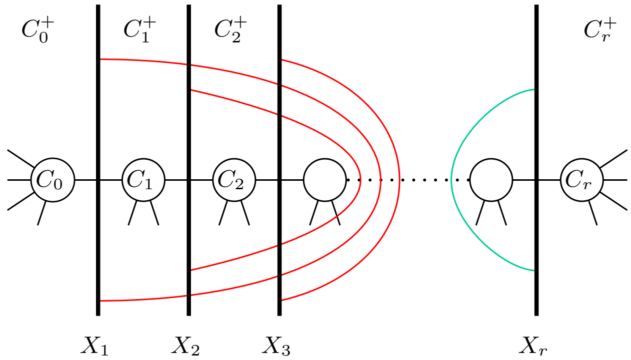

Fig. 4. The path $\tilde{P}$ in the mountain connection tree T of the mountain structure $(G, \varphi, \mathcal{S})$ with the crest separators $X_1, \ldots, X_r$ between the $(\mathcal{S}, \varphi)$-components being nodes of $\tilde{P}$. In addition, 3 right and 1 left pseudo shortcut are shown. If in the figure a curve representing a pseudo shortcut P of a crest separator X encloses a node representing an $(\mathcal{S}, \varphi)$-component C, this should mean that at least some vertices of C are enclosed by the composed cycle of (X, P). Then, by Lemma 19 the vertices of the crest contained in C are enclosed by the composed cycle.

height. But then there is another crest separator $X' = (P_1', P_2')$ in $\mathcal{S}(G, \varphi)$ of the same height than X_i that separates H_{i-1} and H_i and that has only one top vertex . More precisely, P_1' consists of the down path of u . Moreover, for a neighborhood representant v for one of the two neighbors of u on P, P_2' consists of the concatenation of the edge $\{u, v\}$ and the down path of v. Because of Lemma 8 there is only one crest separator in $\mathcal{S}$ going weakly between C_{i-1} and C_i. The existence and properties of X' and the fact that $(G, \varphi, \mathcal{S})$ is a mountain structure implies that X_i cannot be a crest separator in $\mathcal{S}$—contradiction. $\square$

Lemma 20. *Assume that neither X_1 nor X_r has a lowpoint of height at least h, that X_r has no ℓ-long h-high right pseudo shortcut, and that each X_i ($i \in \{2, \ldots, r\}$) has an ℓ-long h-high left pseudo shortcut L_i contained in the $(\{X_1\}, \varphi)$-component D opposite to C_0^+. Then, X_1 cannot have an ℓ-long h-high right pseudo shortcut.*

Proof. First note that no crest separator $X \in \{X_1, \ldots, X_r\}$ can have a lowpoint of height at least h. Otherwise, X would enclose either all $(\mathcal{S}, \varphi)$-components left from X or all $(\mathcal{S}, \varphi)$-components right from X. This would imply that either X_1 or X_r has also a lowpoint of height at least h, which is a contradiction to our assumptions.

Assume for a contradiction that the lemma does not hold, i.e., that there is a strict ℓ-long h-high right pseudo shortcut P_1 of X_1. Define $i \in \{1, \ldots, r\}$ to be as small as possible such that P_1 is contained in the subgraph of G induced by the vertices and edges of the $(\{X_1, \ldots, X_r\}, \varphi)$-components $C_1^+, \ldots, C_i^+$. Since X_r has no ℓ-long h-high right pseudo shortcut, Lemma 13 implies that $i \leq r - 1$. We can also conclude by Lemma 13 that P_1 has exactly one right ℓ-long h-high right pseudo shortcut P_1' of X_i. By Lemma 19, the inner graph of the composed cycle of (X_i, P_1') contains H_i and therefore the same is true for the inner graph of the composed cycle of (X_1, P_1). Similarly, for an arbitrarily chosen strict ℓ-long h-high left pseudo shortcut P_2 of X_{i+1} contained in D, the inner graph of the composed cycle of (X_{i+1}, P_2) also contains H_i. Consequently, P_1 and P_2 must cross at least two times. Since P_1 and P_2 connect their endpoints by shortest paths, one can

easily see that it is possible to interchange subpaths of P_1 with subpaths of P_2 of the same length such that afterwards the inner graphs of the new composed cycles of (P_1, X_1) and (P_2, X_{i+1}) do not intersect anymore except in some common vertices of the old paths P_1 and P_2 and such that the new composed cycles do not enclose the crest H_i anymore. But this is a contradiction to Lemma 19. $\square$

We call a crest separator *ℓ-long h-high pseudo shortcut free* if it has neither an ℓ-long h-high pseudo shortcut nor a lowpoint of height at least h.

Lemma 21. *Assume that X_1 has no ℓ-long h-high left pseudo shortcut, that X_r has no ℓ-long h-high right pseudo shortcut, and that neither X_1 nor X_r has a lowpoint of height at least h. Then one of the crest separators $X_1, \dots, X_r$ is ℓ-long h-high pseudo shortcut free.*

Proof. Note that no crest separator $X \in \{X_1, \dots, X_r\}$ can have a lowpoint of height at least h. Otherwise, X would enclose either all $(\mathcal{S}, \varphi)$-components left from X or all $(\mathcal{S}, \varphi)$-components right from X. This would imply that either X_1 or X_r has also a lowpoint of height at least h, which is a contradiction.

W.l.o.g. all crest separators X_i ($i \in \{2, \dots, r\}$) have an ℓ-long h-high left pseudo shortcut. Otherwise, it suffices to prove the lemma for a shorter sequence of crest separators. Consequently, $X_2, \dots, X_r$ also have a strict ℓ-long h-high left pseudo shortcut. Since X_1 has no ℓ-long h-high left pseudo shortcut, Lemma 13 implies that all strict left h-high pseudo shortcuts of $X_2, \dots, X_r$ are contained in the $(\{X_1\}, \varphi)$-component opposite to C_0^+. We apply Lemma 20 and conclude that X_1 cannot have an ℓ-long h-high right pseudo shortcut. Consequently, X_1 is ℓ-long h-high pseudo shortcut free. $\square$

We next want to introduce a second type of coast separators. Let us define the *inner graph* of a coast separator to be the inner graph of the cycle induced by the vertices of the coast separator. Let us call a coast separator Y to be *h-high* if it consists exclusively of vertices of height at least h. An (h-high) coast separator for a set U is *minimal* (h-minimal) for U if it has minimal size among all (h-high) coast separators for U and if among all such separators its inner graph has a minimal number of inner faces.

The second type of coast separators that we want to consider are the h-minimal coast separators. Therefore, we next study under which conditions an h-minimal coast separator can cross a crest separator or another h-minimal coast separator. The next lemma shows that h-minimal coast separators cannot cross h-high pseudo shortcut free crest separators.

Lemma 22. *Let Q be a cycle in (G, φ) whose vertex set is an h-minimal coast separator Y for a crest H. Moreover, let X be a crest separator in $\mathcal{S}$ that, if it has a lowpoint of height at least h, does not enclose H. Then, for the (X, φ)-component D that does not contain H, Q has at most one maximal (D, X)-subpath, which, if it exists, is a strict h-high D-pseudo shortcut of X.*

Proof. Take $X = (P_1, P_2)$. Assume first that Q contains two (or more) maximal (D, X)-subpaths. See Fig. 5 for an illustration. Thus, for some $i \in \{1, 2\}$, Q contains two vertices s' and s'' on P_i such that the crest path P of X connecting s' and s'' does not belong completely to Q, but is part of the inner graph of Q. Let now Q_1 be the cycle induced by P and the subpath of Q from s'' to s', and let Q_2 be the cycle induced by the reverse path of P and the subpath of Q from s' to s''. Then one of the cycles must enclose H since X contains no vertex of H. By symmetry we assume

w.l.o.g. that this is the case of Q_1. By Lemma 12.a the length of Q_1 cannot be larger than that of Q. However, since P does not completely follow Q, but is part of the inner graph of Q, the inner graph of Q_1 has a smaller number of inner faces than Q. Hence, the vertex set of Q cannot be an h-minimal coast separator—contradiction. If Q contains exactly one maximal (D, X')-subpath P', then, because of Q being an h-minimal coast separator for H and because of H not being contained in D, P' must be a strict h-high D-pseudo shortcut of X. $\square$

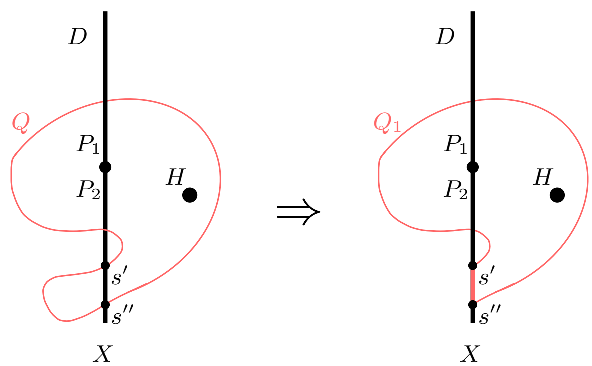

Fig. 5. The replacement of a minimal coast separator Q that crosses a crest separator X more than twice.

Corollary 23. *Let $\mathcal{S}'$ be the set of all ℓ-long h-high pseudo shortcut free crest separators in $\mathcal{S}$, and let H be a crest H of (G, φ) contained in an $(\mathcal{S}', \varphi)$-component C. Then each h-minimal coast separator for H of size at most ℓ consists completely of vertices in C.*

Proof. Let Q be the cycle induced by an h-minimal coast separator of size at most ℓ for H. If Q visits at least one vertex v not contained in C, then there is a crest separator $X \in \mathcal{S}'$ strongly going between $\{v\}$ and H. Since no crest separator in $\mathcal{S}'$ has a lowpoint of height at least h, Q must contain a maximal D-subpath for the (X, φ)-component D containing v. By Lemma 22 this path must be an ℓ-long h-high pseudo shortcut of X, and this is a contradiction to the fact $X \in \mathcal{S}'$. $\square$

An $(\mathcal{S}, \varphi)$-component C is called *ℓ-long h-high pseudo shortcut free* if, for all crest separators $X \in \mathcal{S}$ with a top edge in C and the (X, φ)-component D containing C, there is no ℓ-long h-high D-pseudo shortcut for X and if additionally in the special case, where X encloses C, X does not have a lowpoint of height at least h. The next lemma implicitly shows that, for two crests H' and H'' contained in different ℓ-long h-high pseudo shortcut free $(\mathcal{S}, \varphi)$-components, an h-minimal coast separator of size at most ℓ for H' cannot cross an h-minimal coast separator of size at most ℓ for H''.

Lemma 24. *Let $\mathcal{H}$ be the set of all crests of (G, φ) that are contained in an ℓ-long h-high pseudo shortcut free $(\mathcal{S}, \varphi)$-component, and let $\mathcal{S}'$ be the set of all ℓ-long h-high pseudo shortcut free crest separators in $\mathcal{S}$. Then, for each pair of crests $H', H'' \in \mathcal{H}$, there is a crest separator $X \in \mathcal{S}'$ strongly going between H' and H''.*

Proof . We choose $C_0, \ldots, C_r$ as the $(\mathcal{S}, \varphi)$-components on the path in the mountain connection tree T of $(G, \varphi, \mathcal{S})$ from the $(\mathcal{S}, \varphi)$-component containing H' to the $(\mathcal{S}, \varphi)$-component containing H''. For $i \in \{1, \ldots, r\}$, take X_i as the crest separator with a top edge part of both C_{i-1} and C_i. First note that no crest separator $X \in \{X_1, \ldots, X_r\}$ can have a lowpoint of height at least h. Otherwise, such a crest separator would enclose either C_0 or C_r. This would also imply that X_1 or X_r has a lowpoint of height at least h and encloses C_0 and C_r, respectively. This contradicts our choice of C_0 and C_r as ℓ-long h-high pseudo shortcut free $(\mathcal{S}, \varphi)$-components. Consequently, by Lemma 21, there must be a crest separator $X \in \mathcal{S}'$ going weakly between C_0 and C_r. Moreover, since $\mathcal{S}' \subseteq \mathcal{S}$ and since $(G, \varphi, \mathcal{S})$ is a good mountain structure, X goes also strongly between H' and H''. □

Corollary 25. *Let $\mathcal{S}'$ be the set of all ℓ-long h-high pseudo shortcut free crest separators in $\mathcal{S}$. Then crests in different ℓ-long h-high pseudo shortcut free $(\mathcal{S}, \varphi)$-components are contained in different $(\mathcal{S}', \varphi)$-components.*

Corollary 23 and Corollary 25 imply that h-minimal coast separators of size at most ℓ for crests in different ℓ-long h-high pseudo shortcut free (S, φ)-components cannot cross each other. To find h-high minimal coast separators of size at most ℓ, we use the next lemma.

Lemma 26. *Let $\mathcal{S}'$ be the set of all ℓ-long h-high pseudo shortcut free crest separators in $\mathcal{S}$, and let H be a crest of (G, φ) contained in an $(\mathcal{S}', \varphi)$-component C. Let n be the number of vertices of C. If there is an h-minimal coast separator for H of size at most ℓ in (G, φ), one can find such a coast separator Y in $O(\ell n)$ time such that the cycle induced by the vertices of Y is contained in C.*

Proof. Since C is a shortcut-free component C, by Corollary 23 an h-minimal coast separator for H must be contained in C if it has a size of at most ℓ. To find an h-minimal coast separator of size ℓ, if it exists, let (C', ψ) be the embedded graph obtained from the $(C, \varphi|_C)$ by adding an extra vertex x into the outer face of $(C, \varphi|_C)$ and by inserting edges from x to all vertices of C that are part of a crest separator in $\mathcal{S}'$ with a top edge in C (i.e., that are part of the boundary of C for an appropriate definition of the boundary). Then, we just have to search for a set of vertices of minimal size that in (C', ψ) separates H from x and from all vertices v of C with $h_\varphi(v) \leq h - 1$ and that among all such sets encloses a minimal number of inner faces in (C', ψ) and hence also a minimal number of inner faces in $(C, \varphi|_C)$. Such a set, if it exists, can be found by applying classical network flow techniques in $O(m'\ell)$ time, where m' is the number of edges of C'. Since we added at most $O(n)$ additional edges to the planar graph C, we have $m' = O(n)$. Note explicitly that an h-minimal coast separator of H of size at most ℓ is contained in C, but may contain vertices of the boundary of C which is the reason for the introduction of the vertex x. □

5 Computing coast separators

In this section we show that, for $(2k+1)$-outerplanar graphs, we can construct a set $\mathcal{Y}$ of coast separators whose vertices all have height at least $h = k + 1$ such that each crest of height $2k + 1$ is enclosed by a coast separator $Y \in \mathcal{Y}$ and such that different coast separators in $\mathcal{Y}$ do not cross.

Lemma 27. *Let (G, φ) be an almost triangulated bigraph of treewidth k with φ being $(2k+1)$-outerplanar. Let V be the vertex set of G. Then, in $O(|V|k^3)$ time, one can construct a good*

mountain structure $\mathcal{M} = (G, \varphi, \mathcal{S})$*, the mountain connection tree* T *for* $\mathcal{M}$*, a set* $\mathcal{P}$ *of cycles in* G*, for each* $P \in \mathcal{P}$*, the inner graph* I *of* P *and the corresponding embedding* $\varphi|_I$ *such that the properties below hold.*

(i) *There is a function* m *mapping each* $P \in \mathcal{P}$ *to a non-empty set of* $(\mathcal{S}, \varphi)$*-components such that*
  - *the elements of* $m(P)$ *considered as nodes of* T *induce a connected subgraph of* T*,*[4]
  - *the subgraph of* G *obtained from the union of all* $(\mathcal{S}, \varphi)$*-components in* $m(P)$ *contains the inner graph* I_P *of* P *as a subgraph,*
  - *the set* $m(P)$ *does not have any* $(\mathcal{S}, \varphi)$*-component as an element that is also an element in* $m(P')$ *for a cycle* $P' \in \mathcal{P}$ *with* $P' \neq P$*.*

(ii) *For each crest* H *of height exactly* $2k+1$ *in* (G, φ)*, there is a cycle* $P \in \mathcal{P}$ *with its vertex set being a* $(k+1)$*-high coast separator for* H *of size at most* $3k-1$*.*

In the rest of this section, we prove Lemma 27. Take $h = k + 1$. We start with presenting an algorithm for the computation of the set $\mathcal{P}$ and then prove the correctness of the algorithm. The algorithm works as follows:

We first construct a good mountain structure $(G, \varphi, \mathcal{S})$ with its mountain connection tree in $O(k|V|)$ time (Lemma 9 and Lemma 10). Afterwards, by a simple depth-first search taking $O(|V|)$ time, for each crest separator $X \in \mathcal{S}$ with a lowpoint v of height $h_\varphi(v) \geq h$, we can find the $(\mathcal{S}, \varphi)$-component C that contains the top edge of X and that is enclosed by X, and we define $X_C = X$. Note that each $(\mathcal{S}, \varphi)$-component can be enclosed by at most one crest separator with a top edge in C. For all remaining $(\mathcal{S}, \varphi)$-components C, for which X_C is not already defined, we set $X_C = nil$. We also want to compute a function mapping each $(\mathcal{S}, \varphi)$-component C to the crest H of (G, φ) contained in C. More precisely, for a more practical implementation, we want to compute a mapping CREST that maps each node of T representing an $(\mathcal{S}, \varphi)$-component C to a vertex of the crest contained in C. It is easy to see that CREST and the inverse function CREST^{-1}, which maps (a vertex of) a crest to the $(\mathcal{S}, \varphi)$-component to which it belongs, can be computed by a further depth-first search in $O(|V|)$ time. In $O(|V|k^3)$ additional time, we compute an h-high D-pseudo shortcut set for all crest separators $X \in \mathcal{S}$ and all (X, φ)-components D (Lemma 18). Afterwards it is easy to determine the set $\mathcal{S}'$ of k-long h-high pseudo shortcut free crest separators and the set $\mathcal{H}^+$ of all crests of G that are contained in a k-long h-high pseudo shortcut free $(\mathcal{S}, \varphi)$-component. Because of Lemma 24 different crests of $\mathcal{H}^+$ are contained in different $(\mathcal{S}', \varphi)$-components. We know by Theorem 2 that, for each crest $H \in \mathcal{H}^+$ of height $2k+1$, there is an h-minimal coast separator for H in (G, φ) and that it is of size at most k. Because of Corollary 23 such a coast separator must be completely contained in the $(\mathcal{S}', \varphi)$-component that contains H. Thus, we can compute h-minimal coast separators for all crest separators in $\mathcal{H}^+$ of height $2k+1$ independently by iterating over the $(\mathcal{S}', \varphi)$-components. This takes $O(k|V|)$ total time because of Lemma 26. For each $H \in \mathcal{H}^+$ of height $2k+1$, we put the h-minimal coast separator Y_H constructed for H, or more precisely the cycle P_H induced by the set of vertices of Y_H into $\mathcal{P}$ and compute its inner graph to determine the set of crests in (G, φ) that are enclosed by P_H. This can be easily done for all crests in $\mathcal{H}^+$ of height $2k+1$ in $O(|V|)$ total time.

Let $\mathcal{H}^-$ be the set of crests in (G, φ) that after this step are not already enclosed by a cycle in $\mathcal{P}$ and that do not belong to $\mathcal{H}^+$.[5] We next show how to enclose the crests in $\mathcal{H}^-$ by coast separators.

[4] Note that the nodes of T are identified with $(\mathcal{S}, \varphi)$ components. Thus, $m(P)$ can be considered as a node set of T or as a set of subgraphs of G.

[5] Note that some crests of $\mathcal{H}^+$ are possibly not enclosed by a cycle in $\mathcal{P}$ since they may have a height lower than $2k+1$. By the definition of $\mathcal{H}^-$, we exclude at least all crests in k-long h-high pseudo shortcut free $(\mathcal{S}, \varphi)$-components.

For this purpose, we decompose T into several subtrees of T as follows: First, using the function CREST and spending $O(|V|)$ time we remove all $(\mathcal{S}, \varphi)$-components from T that do not contain a crest of $\mathcal{H}^-$. Moreover, we also remove from T all edges that connect two $(\mathcal{S}, \varphi)$-components for which there is an k-long h-high pseudo shortcut free crest separator $X \in \mathcal{S}$ weakly going between them. This division of T into subtrees can be done in a time linear in the size of T, i.e., also in $O(|V|)$ time. For the forest F obtained—see Fig. 6 for an example—we apply the following Steps (a)-(e) to each tree T^* of F.

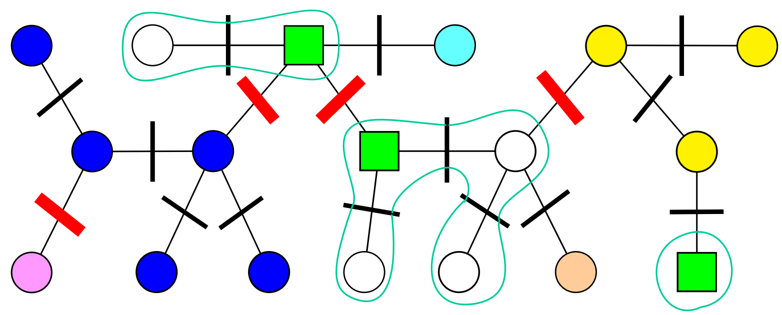

Fig. 6. A mountain connection tree T of the mountain structure $(G, \varphi, \mathcal{S})$ and the crest separators in $\mathcal{S}$; the latter denoted by straight lines crossing the edges of the tree. Different shapes denote different kinds of (S, φ)-components or crest separators. In particular, pseudo shortcut-free components are represented by squared vertices and thick lines mark pseudo shortcut free crest separators. The curves denote coast separators constructed for the crests in the pseudo shortcut free (S, φ)-components. If such a curve in the figure encloses some nodes representing an (S, φ)-component C, this should mean that the corresponding coast separator encloses the crest contained in C. Each set of non-white equal-colored round vertices denotes the nodes of a tree T^* as it is considered by our algorithm.

Intuitively, the Steps (a)-(c) root each tree T^* such that, for each $(\mathcal{S}, \varphi)$-component C not being enclosed by a crest separator, we have a pseudo shortcut P for a crest separator X with a top edge in C such that P uses only vertices that are part of C or of one of the $(\mathcal{S}, \varphi)$-components being descendants of C in T^*. For doing so, the algorithm marks nodes of T^*. Marked nodes do not become the root of T^*. In the Steps (d) and (e), a top-down traversal in T^* is then used to determine coast separators for all crests contained in $(\mathcal{S}, \varphi)$-components that are nodes of T^*.

(a) Initially, consider T^* as undirected tree without any marked nodes, and initialize a set W^* as the set of the $(\mathcal{S}, \varphi)$-components being nodes in T^*. As long as there is a node in W^* with exactly one unmarked neighbor in T^*, run (b). Afterwards proceed with (c).

(b) Take an $(\mathcal{S}, \varphi)$-component $C \in W^*$ with exactly one unmarked neighbor C' in T^*. Delete C from W^* (but not from T^*). Next try to find a cycle P_H enclosing the crest H contained in C as follows: Test if $X_C \neq nil$. If so, mark C, let P_H be the cycle induced by the edges of the essential boundary of X_C, and replace the edge $\{C, C'\}$ by a directed edge (C, C'). Note that P_H encloses H.

Otherwise, take X as the crest separator, whose top edge belongs to C and to C'. Use the precomputed pseudo shortcut sets, and test in $O(1)$ time, whether X has a strict k-long h-high D-pseudo shortcut L for the $(\{X\}, \varphi)$-component D containing C. If and only if such a pseudo shortcut L is found, mark C, define P_H to be the composed cycle of (X, L), and replace the edge $\{C, C'\}$ by a directed edge (C, C'). Also in this case P_H encloses H by Lemma 19.

(c) For each remaining unmarked node C of T^*, do the following: Let H be the crest contained in C. If $X_C \neq nil$, define P_H to be the cycle induced by the edges of the essential boundary of X_C.[6] Otherwise, choose a crest separator $X \in \mathcal{S}$ with a top edge belonging to C such that X has a strict k-long h-high D-pseudo shortcut L for the $(\{X\}, \varphi)$-component D containing C. Note that X indeed exists since C is not a pseudo shortcut free $(\mathcal{S}, \varphi)$-component. In this case, let P_H be the composed cycle of (X, L), which is a cycle enclosing H.

(d) Let $\tilde{T}$ be the graph consisting of the nodes in T^* and the directed edges constructed in Step (b). As we show in Lemma 28, this tree is an *intree*—i.e., a directed tree whose edges are all directed from a node to its parent. Denote the root of $\tilde{T}$ by R. Choose C to be an $(\mathcal{S}, \varphi)$-component of maximal depth such that, for the crest H contained in C, P_H encloses the crest contained in R.[7] As we show in Lemma 29, C can be chosen as the $(\mathcal{S}, \varphi)$-component of maximal depth with X_C being a crest separator going weakly between C and a child of C in $\tilde{T}$, if such an $(\mathcal{S}, \varphi)$-component exists, and as R otherwise. Add P_H to $\mathcal{P}$. Moreover, compute the inner graph $I = (V_I, E_I)$ of P_H and the set $\mathcal{H}_I$ of all crests enclosed by P_H in $O(|V_I|)$ time. This is done by a depth-first search on I starting in a vertex v of H. More precisely, we can choose $v = \mathrm{CREST}(C)$. Initialize now $\mathcal{T}'$ as the forest obtained from $\tilde{T}$ by deleting all $(\mathcal{S}, \varphi)$-components containing a crest in $\mathcal{H}_I$. Using the function CREST^{-1} this can be also done in $O(|V_I|)$ time.

(e) Until all nodes are deleted from $\mathcal{T}'$, repeat: Choose an $(\mathcal{S}, \varphi)$-component C that is a root of one intree in $\mathcal{T}'$. For the crest H contained in C, add P_H to $\mathcal{P}$. Moreover, compute the inner graph $I = (V_I, E_I)$ of P_H and the set $\mathcal{H}_I$ of all crests enclosed by P_H in $O(|V_I|)$ time (similar as in Step (d)). Afterwards, using the function CREST^{-1} delete all $(\mathcal{S}, \varphi)$-components containing a crest of $\mathcal{H}_I$ as nodes from $\mathcal{T}'$.

Concerning the running time, the preprocessing before the Steps (a)-(e) is dominated by the computation of the pseudo shortcut sets, and it therefore runs in $O(k^3|V|)$ time. Let n_{T^*} be the number of nodes in T^*. Then it is easy to see that the running time for the Steps (a)-(c) is bounded by $O(kn_{T^*})$. Since the sum of n_{T^*} over all trees T^* in F is $O(|V|)$, the total running time of the Steps (a)-(c) is $O(k|V|)$. This also holds for the Steps (d) and (e) if no inner face of the inner graph of one of the cycles added to $\mathcal{P}$ is also an inner face of the inner graph of another cycle added to $\mathcal{P}$. We show that this is true in Lemma 32.

For the correctness of the algorithm, we first prove the two lemmata mentioned in Step (d).

Lemma 28. *The graph $\tilde{T}$ considered in Step (d) immediately after its construction is an intree.*

Proof. Let us assume for a contradiction that at some time t_i $(i = 1, 2)$ in Step (b) we consider an $(\mathcal{S}, \varphi)$-component C_i with $X_{C_i} = nil$ and exactly one unmarked neighbor C_i' such that the crest separator X_i weakly going between C_i and C_i' has no k-long h-high D_i-pseudo shortcut for the (X_i, φ)-component D_i containing C_i. Note that at time t_i a neighbor of C_i is only marked if all its neighbors different from C_i are also marked. By induction one can also show that at time t_i

[6] One can also show that this case never occurs.

[7] P_H is then the cycle induced by the essential boundary of a crest separator.

we have already marked all nodes that are reachable from a marked neighbor of C_i by a path not visiting C_i. Since at time t_i the node C_{3-i} is unmarked and C_i' is the only unmarked neighbor of C_i, we can conclude that C_i' lies on the path from C_i to C_{3-i} in T^*. Now, if X_i has a lowpoint of height at least h, then X_i encloses either C_i or C_{3-i}. In the later case, C_{3-i} is also enclosed by X_{3-i}. Hence in both cases, X_i enclosing C_i or X_{3-i} enclosing C_{3-i}, we obtain a contradiction to the fact that $X_{C_1} = X_{C_2} = nil$. Therefore, we can conclude from Lemma 21 that there is a k-long h-high pseudo shortcut free crest separator weakly going between C_1 and C_2, and hence they cannot be both belong to T^*—this means we again have a contradiction. Since we cannot find two $(\mathcal{S}, \varphi)$-components with the properties above, we are able to mark all except one $(\mathcal{S}, \varphi)$-component of T^* so that $\tilde{T}$ is indeed a directed intree with the unmarked node being the root of $\tilde{T}$. $\square$

Lemma 29. *Let $\tilde{T}$ be the tree considered in Step (d), and let C be an $(\mathcal{S}, \varphi)$-component of maximal depth in $\tilde{T}$ with X_C being a crest separator weakly going between C and a child of C, or let $C = R$ if no such $(\mathcal{S}, \varphi)$-component exists. Then, for the crest H contained in C, P_H encloses the crest contained in R. Moreover, for any crest H' contained in an $(\mathcal{S}, \varphi)$-component $C' \neq C$ with C' not being an ancestor of C in $\tilde{T}$ and for the crest separator X' weakly going between C' and its father in $\tilde{T}$, the cycle $P_{H'}$ is completely contained in the (X', φ)-component containing C' and hence does not enclose the crest of R.*

Proof. If, for an $(\mathcal{S}, \varphi)$-component C^* in $\tilde{T}$, X_{C^*} is a crest separator weakly going between C^* and a child of C^*, then the whole (X_{C^*}, φ)-component D^* containing C^* is enclosed by X_{C^*}. In particular, all ancestors of C^* including R are enclosed by X_{C^*}. This proves the first part of the lemma. Let us now choose an $(\mathcal{S}, \varphi)$-component C', a crest H', and a crest separator X' as it is described in the second part of the lemma. If $P_{H'}$ is a composed cycle, then by our construction $P_{H'}$ is the composed cycle of (X', L) for a pseudo shortcut L contained in the (X', φ)-component D' containing C'. In this case, $P_{H'}$ must be completely contained in D'. This is also true if $P_{H'}$ is enclosed by X'. In the remaining case, C' is enclosed by a crest separator that weakly goes between C' and a child of C'. Let C'' be the lowest common ancestor of C and C'. By the observations above, P_H and $P_{H'}$ both must enclose C''. In particular, C'' then must be enclosed by crest separators $\tilde{X}$ and $\tilde{X}'$ weakly going between C'' and that child of C'' that is an ancestor of C and C', respectively. This is a contradiction since an $(\mathcal{S}, \varphi)$-component can be enclosed by at most one crest separator with a top edge in the $(\mathcal{S}, \varphi)$-component. Hence the last case cannot occur. $\square$

Since $(G, \varphi, \mathcal{S})$ is a good mountain structure, the maximal height of a crest separator in $\mathcal{S}$ is $2k$. Therefore, it is easy to see that the size of all coast separators in $\mathcal{P}$ is bounded by $3k - 1$ and that property (ii) of Lemma 27 is true. To prove property (i), we define, for each cycle $P \in \mathcal{P}$, $m(P)$ to be the set of $(\mathcal{S}, \varphi)$-components that contain a crest enclosed by P. The next three lemmata show that property (i) holds.

Lemma 30. *For each $P \in \mathcal{P}$, $m(P)$ induces a connected subgraph of T.*

Proof. Choose an arbitrary but fixed $P \in \mathcal{P}$, and let H be the crest in (G, φ) with $P = P_H$. Assume that there is a crest $H' \neq H$ of (G, φ) that is also enclosed by P_H. Choose an integer $r \geq 1$ and $(\mathcal{S}, \varphi)$-components $C_0, C_1, \ldots, C_r$ such that C_0 and C_r are the $(\mathcal{S}, \varphi)$-components containing H and H', respectively, and such that $C_0, C_1, \ldots, C_r$ are the vertices of the unique simple path from C_0 to C_r in T in the order in which they appear on this path. If P_H is the cycle induced by the edges of the essential boundary of X_{C_0}, then it must enclose an (X_{C_0}, φ)-component D. Since

P_H also encloses H and H', D must contain all $(\mathcal{S},\varphi)$-components $C_0, C_1, \ldots, C_r$. Hence P_H also encloses $C_0, \ldots, C_r$ with their crests.

Otherwise, for $i \in \{1, \ldots, r\}$, we define X_i to be the crest separator $X_i \in \mathcal{S}$ weakly going between C_{i-1} and C_i. Then we can observe that, for all $i \in \{1, \ldots, r\}$, P_H contains a k-long h-high D_i-pseudo shortcut for the (X_i,φ)-component D_i containing C_i. More precisely, this follows from Lemma 13, if P_H is constructed in Step (b) or (c), and from Lemma 22, otherwise. Lemma 19 finally implies that all crests contained in $C_1, \ldots, C_r$ are also enclosed by P_H.

Hence, in both cases, $C_0, C_1, \ldots, C_r \in m(P)$. □

Lemma 31. *For each $P \in \mathcal{P}$, the subgraph of G obtained from the union of all $(\mathcal{S},\varphi)$-components in $m(P)$ contains the inner graph of P as a subgraph.*

Proof. Let H be the crest with $P_H = P$. Assume that there is a vertex v in the inner graph of P_H and that this vertex belongs to no $(\mathcal{S},\varphi)$-component in $m(P_H)$. Let C' be the $(\mathcal{S},\varphi)$-component containing v that in T has the smallest distance to an $(\mathcal{S},\varphi)$-component contained in $m(P_H)$. If P_H is a cycle induced by the edges of the essential boundary of a crest separator X', then because of $C' \notin m(P_H)$, the crest H' contained in C' is not enclosed by X'. Hence v is also not enclosed by P_H, and this is a contradiction to our assumption that v is a vertex of the inner graph of P_H. For the remaining cases, let X be the crest separator with a top edge in C' going weakly between C' and the $(\mathcal{S},\varphi)$-components contained in $m(P_H)$. In particular, $v \notin X$. Hence, P_H must contain a maximal (D', X)-subpath for the (X,φ)-component D' containing C'. Moreover, this subpath must be a k-long h-high D'-pseudo shortcut by Lemma 13, if P_H is constructed in Step (b) or (c), or by Lemma 22, otherwise. Consequently, the crest in C' is enclosed by P_H (Lemma 19). Then we must have $C' \in m(P_H)$, which contradicts our assumptions. □

Lemma 32. *For each $P \in \mathcal{P}$, $m(P)$ does not have any $(\mathcal{S},\varphi)$-component as an element that is also an element of $m(P')$ for a cycle $P' \in \mathcal{P}$ with $P' \neq P$.*

Proof. Let H and H' be different crests of (G,φ) such that P_H and $P_{H'}$ belong to $\mathcal{P}$. We have to show that no $(\mathcal{S},\varphi)$-component belongs to both, $m(P_H)$ and $m(P_{H'})$. This is clear if H and H' belong to $\mathcal{H}^+$ because of the Corollaries 23 and 25.

Let us next consider the case, where $H \in \mathcal{H}^+$, $H' \in \mathcal{H}^-$, and $P_{H'}$ is a cycle induced by the edges of the essential boundary of a crest separator; more exactly of the crest separator $X_{C'}$ for the $(\mathcal{S},\varphi)$-component C' containing H'. In this case, $P_{H'}$ encloses an $(X_{C'},\varphi)$-component D'. D' does not contain H since H is contained in a k-long h-high pseudo shortcut free component C.[8] Hence $X_{C'}$ goes weakly between C' and C. Since $m(P_H)$ induces a connected subtree in T (Lemma 30), $m(P_H)$ and $m(P_{H'})$ having a common $(\mathcal{S},\varphi)$-component would imply that C' is contained in $m(P_{H'})$. This is a contradiction to the fact that $H' \in \mathcal{H}^-$.

Note that in all remaining cases with $H \in \mathcal{H}^+$ and $H' \in \mathcal{H}^-$ no crest separator $X \in \mathcal{S}$ weakly going between H and H' can have a lowpoint with a height of at least h since, otherwise, either H or H' would be enclosed by a crest separator with a lowpoint of height at least h. Let us consider the unique simple path P in T from the $(\mathcal{S},\varphi)$-component C containing H to the $(\mathcal{S},\varphi)$-component C' containing H'. Let X be the crest separator going weakly between C and the next vertex on P, and let X_1 be the crest separator going weakly between the last $(\mathcal{S},\varphi)$-component on P containing

[8] Here we use the fact that a k-long h-high pseudo shortcut free (S,φ)-component by definition cannot be enclosed by a crest separator with a low-point of height at least h.

a crest enclosed by P_H and the next vertex on P. We then can apply Lemma 20 with $X_r = X$ to show that X_1 has no k-long h-high D-pseudo shortcut for the (X_1, φ)-component D containing C. Thus, by Lemma 13 $m(P_{H'})$ cannot contain any $(\mathcal{S}, \varphi)$-component that is contained in D, and hence no $(\mathcal{S}, \varphi)$-component contained in $m(P_{H'})$ is also contained in $m(P_H)$.

If H and H' both belong to $\mathcal{H}^-$, but to different subtrees of the forest F defined before the Steps (a)-(e), then either there is crest separator $X \in \mathcal{S}'$ going weakly between H and H' or the simple unique path in T between the two $(\mathcal{S}, \varphi)$-components containing H and H', respectively, contains an $(\mathcal{S}, \varphi)$-component C^* with a crest H^* enclosed by a cycle $P_{H^+} \in \mathcal{P}$ for a crest $H^+ \in \mathcal{H}^+$. Let us first consider the case, where X exists. Let D be the (X, φ)-component containing H. Note that H' is contained in the (X, φ)-component $\tilde{D}$ opposite to D. We show that all $(\mathcal{S}, \varphi)$-components in $m(P_H)$ are contained in D. Then, by an analogous argument all $(\mathcal{S}, \varphi)$-components in $m(P_{H'})$ are contained in $\tilde{D}$, i.e., $m(P_H)$ and $m(P_{H'})$ contain no common $(\mathcal{S}, \varphi)$-component. If P_H is induced by the edges of the essential boundary of a crest separator X', then the crests enclosed by P_H are exactly the crests that are contained in the (X', φ)-component D' enclosed by X'. Since D' does not contain any vertices of height smaller than h, D' is not the (X', φ)-component that contains all vertices of X, i.e., D' is contained in D. Therefore, all $(\mathcal{S}, \varphi)$-components of $m(P_H)$ are completely contained in D. This also holds if P_H is a composed cycle because of Lemma 13 and of the fact that X is k-long h-high pseudo shortcut free.

We next show that P_H and $P_{H'}$ also cannot share a common $(\mathcal{S}, \varphi)$-component if we have an $(\mathcal{S}, \varphi)$-component C^* as described above. This is true because $m(P_H)$ and $m(P_{H'})$ are connected with respect to T by Lemma 30 and because we have already shown that neither $m(P_H)$ nor $m(P_{H'})$ can contain an $(\mathcal{S}, \varphi)$-component whose crest is enclosed by a cycle $P_{H^+} \in \mathcal{P}$ for a crest $H^+ \in \mathcal{H}^+$.

Finally, let H and H' contained in the same subtree T^* of F. Lemma 29 implies that after Step d), for each crest H^* contained in an $(\mathcal{S}, \varphi)$-component C^* in the forest $\mathcal{T}'$, the cycle P_{H^*} is completely contained in C^* and the $(\mathcal{S}, \varphi)$-components being descendants of C^* in $\tilde{T}$. Therefore the top-down construction process in the Steps (d) and (e) guarantees that the sets $m(P_H)$ and $m(P_{H'})$ cannot share a common $(\mathcal{S}, \varphi)$-component. See Fig. 7. $\square$

6 A tree decompositions for the components

We first sketch an algorithm of Bodlaender [6] for constructing a tree decomposition of width $3\ell - 1$ for a graph G with an ℓ-outerplanar embedding φ. Whereas the original algorithm works on general ℓ-outerplanar graphs, we will describe a slightly modified version that is only correct if the instance (G, φ) is a mountain. Let $\mathcal{S} \subseteq \mathcal{S}(G, \varphi)$. The modified version has some nice properties that later help us to solve the main problem of this section; namely, given an $(\mathcal{S}, \varphi)$-component C, to construct a tree decomposition for $\mathrm{ext}(C, \mathcal{S})$ with the following property: For each crest separator $X \in \mathcal{S}$ with a top edge in C, there is a bag containing all vertices of X. This is trivial if $\ell = 1$ since in this case every tree decomposition has this property. This means that, for $\ell = 1$, we can just use the original algorithm of Bodlaender to satisfy our special property.

Let us now consider a fixed ℓ-outerplanar mountain (G, φ) with n vertices for some $\ell \geq 2$. For simplicity, let us assume for a moment that the following *degree constraint* holds for G: Each vertex of v has degree at most 4 and each vertex of degree exactly 4 has exactly one neighbor of height $h_\varphi(v) - 1$. Note that this implicitly implies that each vertex v of G has at most one vertex of height $h_\varphi(v) + 1$. Let G_i be the subgraph of G induced by all vertices of height at least i. The edges of G_i that are incident to the outer face of $\varphi|_{G_i}$ are called the *outer* edges of G_i, whereas the other edges

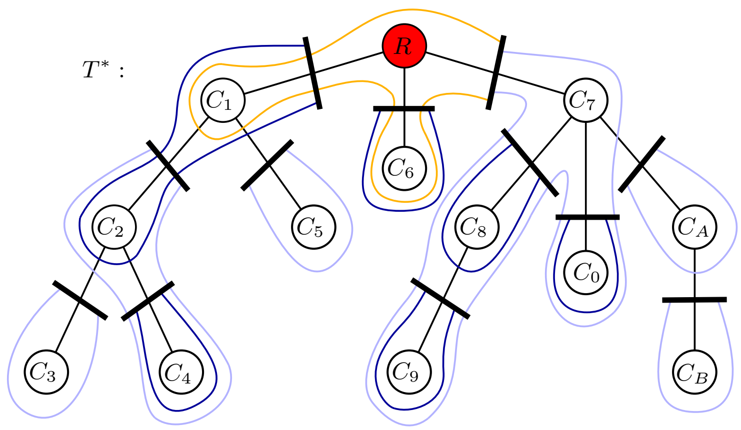

Fig. 7. The construction of coast separators on the subtree T^*, where for simplicity all constructed coast separators are composed cycles. A straight line crossing an edge of the tree denotes a crest separator weakly going between the (S, φ)-components represented by the endpoints of the edge. A curve beginning and ending at a crest separator represents a pseudo shortcut of this crest separator. If in the figure a composed cycle encloses a node representing an $(\mathcal{S}, \varphi)$-component C, this should mean that the composed cycle encloses at least the crest contained in C. The figure shows all composed cycles constructed for crests contained in the $(\mathcal{S}, \varphi)$-components being nodes of T^*. If such a cycle is added to $\mathcal{P}$, the corresponding pseudo shortcut is colored with a bright color. Note that each node of T^* is enclosed by a composed cycle with a bright colored pseudo shortcut.

are the *inner* edges of G_i. In a first step, we construct a spanning tree T for G with the following additional property: For each $i \geq 1$, the subgraph T_i of T induced by the vertices v with $h_\varphi(v) \geq i$

(a) is a spanning tree for G_i.

(b) has as many inner edges as possible among all possible choices such that (a) holds for all $j \geq i$.

Let us call a spanning tree for (G, φ) with this property to be *up-connected*. Clearly, if (G, φ) is a mountain, an up-connected spanning tree exists and can easily be constructed with a modified breadth-first search in $O(n)$ time. The difference between our version and the original version of Bodlaender's algorithm is exactly that we use an up-connected spanning tree. Take $G = (V, E)$ and $T = (V, F)$. Then a *standard tree decomposition* (T^*, B^*) *for* G *with skeleton* T can be constructed as follows:

The node set of T^* is $V \cup F$. The arc set of T^* is obtained by inserting two arcs $\{u_1, e\}$ and $\{u_2, e\}$ for each edge $e = \{u_1, u_2\}$ of T into T^*. For each edge $e = \{u, v\}$ in $E \setminus F$, let us call the cycle consisting of edge e and the unique path from u to v in T^* to be the *fundamental cycle* for e. We define B^* in two steps: First, for each vertex $v \in V$ and each edge $e = \{u_1, u_2\}$ of F, add v into $B^*(v)$, and add u_1 and u_2 into $B^*(e)$. Second, for each edge $\{u_1, u_2\} \in E \setminus F$, add arbitrarily one vertex of $\{u_1, u_2\}$ into the bags of all nodes in T^* that are part of the fundamental cycle for $\{u_1, u_2\}$.

Lemma 33. *For an up-connected spanning tree $T = (V, F)$ of (G, φ) and $d = \max(3, \{deg_T(v) \mid v \in V\})$, the width of a standard tree decomposition (T^*, B^*) for G with skeleton T is at most $d\ell - 1$.*

Proof. One can easily see that (T^*, B^*) has all properties of a tree decomposition for G. All we have to prove is that (T^*, B^*) has width $d\ell - 1$. For each edge $e \in F$ and for each vertex $u \in V$, let $\mu(e)$ and $\nu(u)$ be the number of fundamental cycles for the edges in $E \setminus F$ that contain e or u, respectively. Define $\mu = \max\{\mu(e) \mid e \in F\}$ and $\nu = \max\{\nu(u) \mid u \in V\}$. By the construction of B^* it is easy to see that the width of (T^*, B^*) is bounded by $\max\{\mu + 1, \nu\}$.

For $i \in \{1, \ldots, \ell\}$, let E_i be the edges of G_i contained in $E \setminus F$ with at least one endpoint having height i. Then, $E \setminus F = E_1 \cup \cdots \cup E_\ell$. We next want to analyze, for a fixed $i \in \{1, \ldots, \ell\}$, how the insertion of one endpoint for each edge $e \in E_i$ can increase μ and ν. Note that by our degree constraint each vertex of height i is incident to at most one inner edge e of G_i and that in this case e must be contained in F by property (b). Hence we can conclude that the edges in E_i are outer edges with both endpoints having height i. Since the subgraph of T induced by the vertices v with $h_\varphi(v) \geq i$ is a spanning tree of G_i, each edge $e \in E_i$ encloses an inner face in the graph $G_i^* = (V_i, F \cup E_i)$. Since all edges in E_i are outer edges for G_i, no other edge of E_i can be enclosed by the fundamental cycle for e. Analogously e is not contained in a fundamental cycle for another edge in E_i. Consequently, each inner face of $(G_i^*, \varphi|_{G_i^*})$ is incident to at most one edge $e \in E_i$. Moreover, each edge $e \in F$ can be incident to at most two inner faces of $(G_i^*, \varphi|_{G_i^*})$. Consequently, for a fixed edge $e \in F$, the insertion of endpoints for the edges in E_i, i.e., of one endpoint for each edge in E_i, can increase $\mu(e)$ by at most 2.

Since the degree of all vertices in T is bounded by d, all vertices in T are incident to at most d inner faces of $(G_i^*, \varphi|_{G_i^*})$. This means that, for each $u \in V$, $\nu(u)$ can increase by at most d by the insertion of endpoints for the edges of E_i. Moreover, in the special case $i = \ell$, G_ℓ^* is outerplanar. Thus, each vertex of T can be incident to at most $d - 1$ inner faces of $(G_\ell^*, \varphi|_{G_\ell^*})$. The insertion of endpoints for the edges in E_ℓ can therefore increase $\nu(u)$ by at most $d-1$ for each $u \in V$. Altogether, we can conclude that the treewidth is bounded by at most $\max\{\mu + 1, \nu\} = \max\{2\ell + 1, d\ell - 1\} = d\ell - 1$, where the last equality follows from $\ell \geq 2$ and $d \geq 3$. $\square$

To obtain a tree decomposition of width $3\ell - 1$, the algorithm of Bodlaender bounds the degree of each vertex in G by splitting each vertex v of larger degree into a path of vertices of degree 3 as it is sketched in Fig. 8. More precisely, for each vertex v of degree at least four, this can be done in the following way: Let F be an inner face of (G, φ) whose boundary contains v and a vertex of height $h_\varphi(v) - 1$, or let F be the outer face if $h_\varphi(v) = 1$. Let $\{v, u_1\}, \ldots, \{v, u_d\}$ be the edges incident to v in the clockwise order in which their appear around v with $\{v, u_1\}$ and $\{v, u_d\}$ being part of the boundary of F. Then v is replaced by a series of vertices $v_1, \ldots, v_{d-2}$ with v_1 being incident to u_1, u_2, and v_2, whereas v_{d-2} is incident to v_{d-3}, u_{d-1}, u_d and, for $i \in \{2, \ldots, d-3\}$, v_i is incident to v_{i-1}, u_{i+1}, and v_{i+1}. The vertices $v_1, \ldots, v_{d-2}$ are then called the *copies of* v. A vertex v of degree at most 3 in G is not split and also called to be a *copy of* v. It is not hard to see that a replacement as described above can be done for all vertices of degree at least 4 in such a way that afterwards we obtain an embedded graph (G^*, φ^*) for which the following holds:

(1) $h_{\varphi^*}(v^*) = h_\varphi(v)$ for each copy v^* of a vertex v,
(2) for two vertices u and v on boundary of one inner face, there are copies u^* of u and v^* of v and an inner face F in (G^*, φ^*) such that u^* and v^* are on the boundary of F.

The algorithm of Bodlaender continues with constructing an up-connected spanning tree T of G^*. Clearly, for each vertex v of G^*, we have $\deg_T(v) \le 3$ since $\deg_{G^*}(v) \le 3$. Afterwards a standard tree decomposition (T^*, B^*) for G^* of width $3\ell - 1$ is constructed by applying Lemma 33. Replacing each copy of a vertex in a bag of (T^*, B^*) by the original vertex finally yields to a tree decomposition of width $3\ell - 1$ for G. It is not hard to see that the replacement of (G, φ) by (G^*, φ^*), the construction of a tree decomposition for G^*, and the final replacement of copies by original vertices can be done in $O(n\ell)$ time.

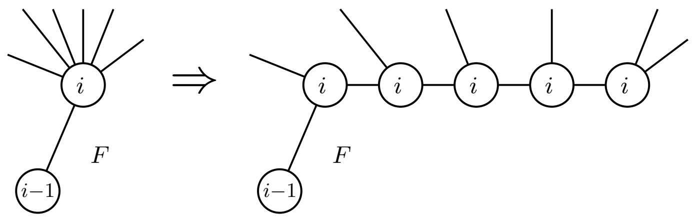

Fig. 8. The replacement of a vertex of high degree in Bodlaender's algorithm. The vertices are labeled with their height.

Theorem 34. *Let $\ell \in I\!N$, and let $M = (G, \varphi, \mathcal{S})$ be a good mountain structure with an ℓ-outerplanar embedding φ. Let C be an $(\mathcal{S}, \varphi)$-component, and let $C' = \mathrm{ext}(C, \mathcal{S})$. Then one can construct a tree decomposition (T, B) of width $3\ell - 1$ for C' that, for each crest separator $X \in \mathcal{S}$ with a top edge in C, has a bag containing all vertices of X. Moreover, given C' as well as the heights $h_\varphi(v)$ for each vertex v in C', the construction can be done in $O(n\ell^2)$ time, where n is the number of vertices in C'.*

Proof. W.l.o.g. $\ell \ge 2$. We first show that the case, where there is a crest separator $X \in \mathcal{S}$ enclosing C, can be reduced to the case, where this is not the case. If X exists, let v be the lowpoint of X and $i = h_\varphi(v)$. Then we simply remove all vertices u with $h_\varphi(u) \le i$ from G, from C, and from the crest separators contained in $\mathcal{S}$, and additionally we remove from $\mathcal{S}$ all crest separators that afterwards have no vertices anymore. We then obtain a new good mountain structure $(G', \varphi', \mathcal{S}')$, where φ' is an $(\ell - i)$-outerplanar embedding. More precisely, if the graph G after removing all vertices of height at most i is not biconnected, we obtain a good mountain structure for each biconnected component. After the modifications no crest separator in $\mathcal{S}'$ encloses the new $(\mathcal{S}', \varphi')$-component obtained from C by deleting v. Thus, if our lemma holds for this case, it can be applied to construct a tree decomposition of width $3(\ell - i) - 1$ for G' that, for each crest separator $X' \in \mathcal{S}$ with a top edge in C, has a bag containing all vertices u of X' with $h_\varphi(u) > i$. Since C is enclosed by X, the remaining vertices of X' are all part of the down path of v in (G, φ). We can simply add these remaining i vertices of the down path of v into all bags of the tree decomposition. Hence, from now on we assume that C is not enclosed by a crest separator in $\mathcal{S}$. For simplicity, we also want to assume that all crest separators in $\mathcal{S}$ have two top vertices and that they do not have a lowpoint. At the end of our proof we will show how to handle the other types of crest separators.

The heights of the vertices in C' with respect to $\varphi|_{C'}$ may differ from the heights of these vertices with respect to φ. In order to avoid this and to be better prepared for the later steps of our

algorithm, we insert additional edges into C'. More precisely, for each crest separator $X = (P_1, P_2)$ with a top edge in C and for the vertices $u_1, \ldots, u_q$ of P_1 and $v_1, \ldots, v_q$ of P_2 in the order of their appearance on these paths, we add a new vertex x_i and edges $\{u_i, x_i\}$ and $\{x_i, v_i\}$ into the outer face of $\varphi|_{C'}$ for all $i \in \{1, \ldots, q\}$. This results in a graph $\tilde{C}$ with a unique combinatorial embedding $\tilde{\varphi}$ for which $h_{\tilde{\varphi}}(v) = h_{\varphi}(v)$ holds for all vertices v of C'. Moreover, if $u_1, \ldots, u_q, v_1, \ldots, v_q, x_1, \ldots, x_q$ are defined as above, we have that x_i has the same height as u_i and v_i in $\tilde{\varphi}$ since $h_{\tilde{\varphi}}(x_i) = q-i+1 = h_{\varphi}(u_i) = h_{\tilde{\varphi}}(u_i) = h_{\tilde{\varphi}}(v_i)$. From now on we try to construct a tree decomposition for $\tilde{C}$. We call the newly introduced vertices *virtual vertices* and the new edges incident to these vertices *horizontal virtual edges*.

As in Bodlaender's algorithm we first replace $(\tilde{C}, \tilde{\varphi})$ by a new graph (C^*, φ^*) with all vertices in C^* having degree 3 such that the properties (1) and (2) hold. Let us consider a fixed crest separator $X = (P_1, P_2)$ in $\mathcal{S}$ with a top edge in C. Let $u_1, \ldots, u_q$ be the vertices of P_1 and $v_1, \ldots, v_q$ be the vertices of P_2 in the order in which they appear on P_1 and P_2, respectively. Moreover, for $i \in \{1, \ldots, q\}$, let x_i be the virtual vertex adjacent to u_i and v_i in $\tilde{C}$. The replacement of $\tilde{C}$ by C^* replaces each virtual vertex by a single copy whereas the other endpoints of the horizontal virtual edges might by replaced by several copies. For $i \in \{1, \ldots, q\}$, we use u_i^* to denote the copy of u_i in C^* with $\{u_i^*, x_i\}$ being an edge of C^*. The vertex v_i^* is defined analogously. Similarly, we use u_1' and v_1' to denote the copies of u_1 and v_1, respectively, with $\{u_1', v_1'\}$ being an edge of C^*. This edge must exist since $\{u_1, v_1\}$, as the top edge of X, is an edge in G. To guarantee that a tree decomposition for C^* has a bag that, for each vertex $v \in X$, contains a copy of v, we apply some extra modifications. Note that the insertion of an edge e into an inner face of an embedded graph does not change the height of any vertex if the endpoints of e have different height. Since $h_{\varphi}(x_i) \neq h_{\varphi}(x_{i+1})$ for all $i \in \{1, \ldots, q-1\}$ and because of the properties (1) and (2), we can add further edges $\{x_1, x_2\}, \ldots, \{x_{q-1}, x_q\}$ to C^* without changing the height of any vertex or destroying the planarity. These edges are called *vertical virtual edges*. Let (C^+, φ^+) be the graph that we obtain after applying the above changes to all crest separators in $\mathcal{S}$ with a top edge in C. Define C^- to be the graph obtained from C^+ by removing all virtual vertices and their incident edges. Since, for each vertex v of G part of a crest separator, there is a path in G from a top vertex u of the crest separator to v not visiting any vertex of height lower than $h_{\varphi}(v)$, there are paths from each copy of u to each copy of v in C^- such that the paths do not visit any vertex of height lower than $h_{\varphi}(v) = h_{\varphi^+}(v)$. Therefore, one can easily construct a spanning tree T^- for C^- such that, for each $i \in I\!N$, the subgraph of T^- induced by the vertices v with $h_{\varphi}(v) \geq i$ is a spanning tree for the subgraph of C^- induced by the vertices v with $h_{\varphi}(v) \geq i$. In particular, we can choose T^- such that among all possible choices, for each $i \in \{1, \ldots, \ell\}$, it uses a maximal number of edges not incident to the outer face in the subgraph of C^+ induced by edges of height at least i. If we add to T^- the path consisting of the edges $\{u_1^*, x_1\}, \{x_1, x_2\}, \ldots, \{x_{q-1}, x_q\}$ and if we run a similar path addition for all other crest separators, we obtain an up-connected spanning tree T for (C^+, φ^+). Moreover, since the degree of each non-virtual vertex in C^+ is at most 3, we have $\deg_T(v) \leq 3$ for all vertices v in C^+. We now construct a standard tree decomposition (T^*, B^*) for C^+ with skeleton T, but whenever we insert one endpoint of a horizontal virtual edge e into the bags of the fundamental cycle for e, we always choose the non-virtual vertex for these insertions. For $\tilde{e} = \{u_1^*, x_1\}$, $B^*(\tilde{e})$ contains copies of all vertices of X. Since T is an up-connected spanning tree where the degree of each vertex is bounded by 3, the width of (T^*, B^*) is at most $3\ell - 1$ by Lemma 33. After replacing in each bag of (T^*, B^*) the copies by the original vertices in C', we obtain a tree decomposition for C' with a bag containing all vertices of X. Since we have applied our modifications also to all

other crest separators in $\mathcal{S}$ with a top edge in C, for each such crest separator X', there is also a bag containing the vertices of X'.

We next give a short sketch how to handle the other types of crest separators. The idea is always to apply some extra changes to $(\tilde{C}, \tilde{\varphi})$ before we start with replacing $(\tilde{C}, \tilde{\varphi})$ by (C^*, φ^*). Let us first consider a crest separator $X = (P_1, P_2)$ that has only one top vertex, and let $u_1, \ldots, u_q$ and $v_1, \ldots, v_q$ be the vertices of P_1 and P_2, respectively, in the order of their appearance on these paths. See Fig. 9. Let F be the inner face in $(\tilde{C}, \tilde{\varphi})$ that is incident to the boarder edge $\{u_1, u_2\}$. Then we replace the edge $\{u_1, v_2\}$ by a new vertex v_1' embedded into F and new edges $\{u_1, v_1'\}$ and $\{v_1', v_2\}$. This does not change the height of any of the old vertices and $h_{\tilde{\varphi}}(v_1') = h_{\tilde{\varphi}}(u_1)$. If we replace the vertex v_1 in P_2 by v_1', we can afterwards handle X as a crest separator with two top vertices.

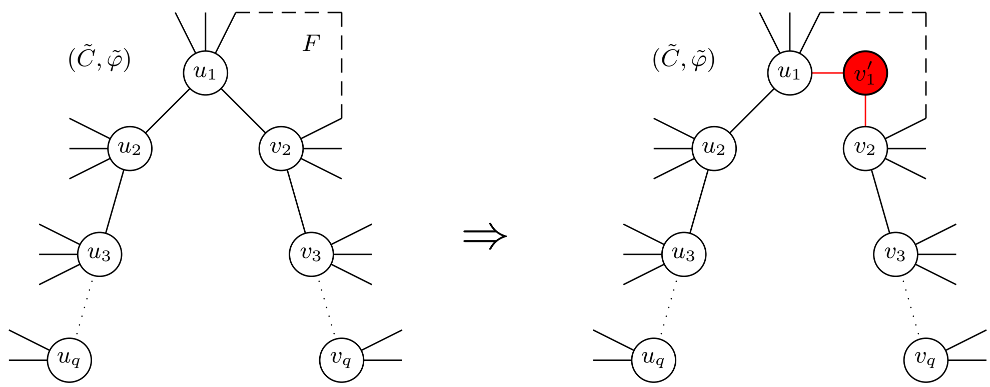

Fig. 9. The replacement of a crest separator with one top vertex $u_1 = v_1$ by a crest separator with two top vertices u_1' and v_1.

Let us next consider a crest separator $X = (P_1, P_2)$ with a lowpoint. Define $u_1, \ldots, u_q$ and $v_1, \ldots, v_q$ as the vertices of P_1 and P_2 in the order in which they appear on P_1 and P_2, respectively. Assume that $u_j = v_j$ is the lowpoint of X. Since we want to use a similar construction as for crest separators without any lowpoints, the idea is, for all $i \geq j$, to split u_i in $\tilde{C}$ into two different vertices u_i' and v_i' and to connect them by a path consisting of two edges $\{u_i', x_i\}$ and $\{x_i, v_1'\}$ for a new virtual vertex x_i. Moreover, $u_j', \ldots, u_q'$ should be connected to the neighbors of P on one side and the vertices $v_j', \ldots, v_q'$ to the neighbors of P on the other side of P. Additionally, each edge $\{u_i, u_{i+1}\}$ $(i = j, \ldots, q-1)$ is split into two edges $\{u_i', u_{i+1}'\}$ and $\{v_i', v_{i+1}'\}$. Then, the tuple of the two new paths visiting $u_1, \ldots, u_{j-1}, u_j', \ldots, u_q'$ and $v_1, \ldots, v_{j-1}, v_j', \ldots, v_q'$, respectively, can be considered as a crest separator without a lowpoint. See Fig. 10. If a vertex is a lowpoint of several crest separators, note that we have to do this splitting for each of the crests separators. It is not hard to see that the splitting process for different crest separators does not lead to a non-planar embedding as long as there is no pair of crest separators X_1 and X_2 in $\mathcal{S}$ with top edges in C for which the vertices of the essential boundary of X_1 are enclosed by X_2. Then C' must be enclosed by one of X_1 and X_2, but we have already excluded this case at the beginning of our proof.

The above splitting process can add for each crest separator only $O(\ell)$ additional edges and vertices. This is the reason for the additional factor ℓ in the running time in comparison to Bodlaender's algorithm. $\square$

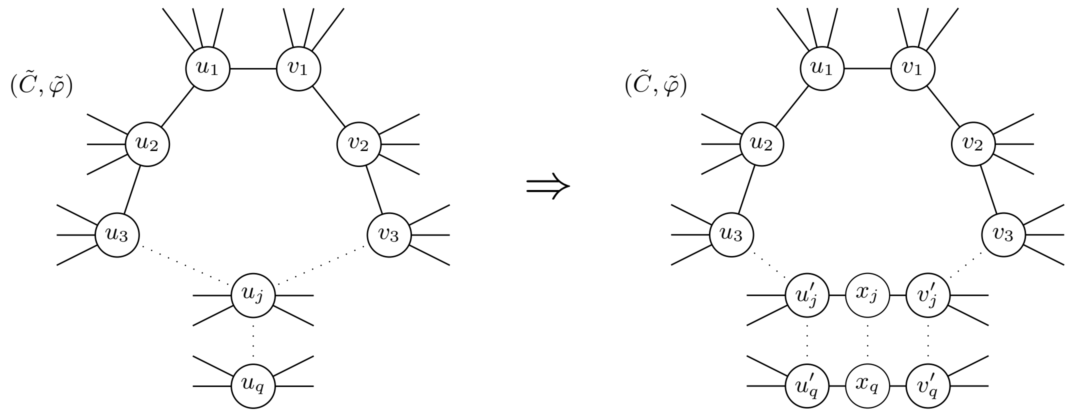

Fig. 10. The replacement of a crest separator with a lowpoint by a crest separator with no lowpoint.

7 The main algorithm

In this section we describe our main algorithm. As mentioned in Section 3, we assume that we are given an almost triangulated graph (G, φ) of treewidth k. We also assume that G is biconnected since, otherwise, we can independently construct a tree decomposition for each biconnected component, and we can combine these tree decompositions by well-known techniques to a tree decomposition for G. Our algorithm starts with cutting off all maximal connected subsets of vertices of height at least $h = 2k + 1$ by coast separators of small size. More precisely, to find such coast separators, in a first substep we merge each maximal connected set M of vertices of height at least h to one vertex v_M so that the graph (G', φ') obtained is $(2k + 1)$-outerplanar. Given a vertex of the coast, this can be done in a time linear in the number of vertices with a height of at most $h+1$. G' is an almost triangulated bigraph since this is true for G. We therefore can use Lemma 10 to construct a good mountain structure $(G', \varphi', \mathcal{S})$ and apply Lemma 27 to find a set of cycles $\mathcal{P}$ and a function m that maps the cycles to $(\mathcal{S}, \varphi')$-components such that the following properties of the lemma hold.

- each crest of height h is enclosed by a cycle $P \in \mathcal{P}$.
- each cycle $P \in \mathcal{P}$ has length at most $3k - 1$ and all its vertices are of height at least $k + 1$.
- for each $(\mathcal{S}, \varphi')$-component C, there is at most one $P \in \mathcal{P}$ with $C \in m(P)$.
- the inner graph of a cycle $P \in \mathcal{P}$ is a subgraph of the graph obtained from the union of the $(\mathcal{S}, \varphi')$-components in $m(P)$.
- the $(\mathcal{S}, \varphi')$-components in $m(P)$ induce a connected subgraph of the mountain connection tree.

Since G' is h-outerplanar, we can apply Theorem 34 to each $(\mathcal{S}, \varphi')$-component C to compute a tree decomposition (T_C, B_C) of width at most $3h-1$ for $\mathrm{ext}(C, \mathcal{S})$ such that, for each crest separator $X \in \mathcal{S}$ with a top edge in C, (T_C, B_C) has a node whose bag contains all vertices of X. This node is then connected to a node whose bag also contains all vertices of X and that is constructed for $\mathrm{ext}(C', \mathcal{S})$ with C' being the other $(\mathcal{S}, \varphi')$-component C' containing the top edge of X. Since the set of common vertices of $\mathrm{ext}(C, \mathcal{S})$ and $\mathrm{ext}(C', \mathcal{S})$ is a subset of the vertices of X, after also connecting nodes for all other crest separators in $\mathcal{S}$, we obtain a tree decomposition (T^*, B^*) for G'.

Let us next remove from $\mathcal{S}$ all crest separators whose top edge is contained in two $(\mathcal{S}, \varphi')$-components belonging to the same set $m(P)$ for some $P \in \mathcal{P}$. Afterwards, for the new set $\mathcal{S}'$

of crest separators, each cycle $P \in \mathcal{P}$ is contained in one $(\mathcal{S}', \varphi')$-component.[9] For each $(\mathcal{S}', \varphi')$-component C', let us call the *flat component* of C' to be the subgraph of $\text{ext}(C', \mathcal{S}')$ obtained by removing the vertices of the inner graph of the cycle $P \in \mathcal{P}$ with P contained in C' if such a cycle P exists. If not, we define the *flat component* to be $\text{ext}(C', \mathcal{S}')$ which then contains no vertex of height h. We then remove from the bags in (T^*, B^*) all vertices that do not belong to a flat component. Afterwards, for each cycle $P \in \mathcal{P}$ disconnecting the crests of an $(\mathcal{S}', \varphi')$-component C' from the coast, we put the vertices of P into all bags of the tree decompositions (T_C, B_C) constructed as part of (T^*, B^*) for the (extended components of the) $(\mathcal{S}, \varphi')$-components C contained in C'. This allows us to connect one of these bags with a bag of a tree decomposition for the inner graph of P. Indeed, for each $P \in \mathcal{P}$, we recursively construct a tree decomposition (T_P, B_P) for the inner graph G_P of P with the vertices of P being the coast of G_P. Into all bags of that tree decomposition that are not constructed in further recursive calls, we also put the vertices of P. At the end of the recursions, we obtain a tree decomposition for the whole graph. Note that each bag is of size $O(k)$. More precisely, let us consider a recursive call that constructs a tree decomposition (T_P, B_P) for a cycle P constructed in a previous step. Then the tree decomposition for the flat component considered in the current recursive call will put up to $3h = 6k + 3$ vertices into each bag. However, we also have to insert the vertices of P and possibly the vertices of a cycle constructed in the current recursive call into the bags. Since each of these cycles consists of at most $3k - 1$ vertices, each bag of the final tree decomposition of G contains at most $12k + 1$ vertices.

Concerning the running time, it is easy to see that each recursive call is dominated by the computation of the cycles being used as coast separators. This means that each recursive call runs in $O(nk^3)$ time, where n is the number of vertices of the subgraph of G considered in this call. Some vertices part of one recursive call are cut off from the current graph and then are also considered in a further recursive call. However, since the cycles used as coast separators consists exclusively of vertices of height at least $k + 1$, a vertex is considered in at most two recursive calls. Therefore, given a planar graph $G = (V, E)$ of treewidth k, our algorithm finds a tree decomposition for G of width $O(k)$ in $O(|V|k^3)$ time. If we do not know k in advance, we can use a binary search to determine a tree decomposition for G of width $O(k)$ in $O(|V|k^3 \log k)$ time.

Theorem 35. *For a planar graph G of treewidth k with n vertices, a tree decomposition of width $O(\text{tw}(G))$ can be constructed in $O(nk^3 \log k)$ time.*

More precisely, we have shown that within the time bound from above one can find a tree decomposition of width $12\text{tw}(G) + 1$ if G is almost triangulated. As mentioned in Section 3, it is possible to replace a non-triangulated graph G by an almost triangulated graph G' of treewidth $4\text{tw}(G)+1$ so that we obtain a tree decomposition of width $48\text{tw}(G)+13$. However, this is not really necessary. Using some more sophisticated techniques, we can obtain a tree decomposition of width $12tw(G) - 9$ for general planar graphs. More generally, for any $0 < \alpha \leq 1$, we can also show [13] that, for a planar graph G of treewidth k with n vertices, one can construct a tree decomposition of width $9k + 3\lfloor \alpha k \rfloor + 9$ in $O(n \min\{1/\alpha, k\} k^3 \log k)$ time.

References

1. E. Amir. Approximation algorithms for treewidth. *Algorithmica*, 56(4):448–479, 2010.

[9] We now have found a set of crest separators and coast separators that guarantee (P1) - (P3) from page 4. The set $\mathcal{S}'$ is exactly the set of perfect crest separators.

2. S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.
3. S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991.
4. H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernet.*, 11(1–2):1–21, 1993.
5. H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
6. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoret. Comput. Sci.*, 209(1–2):1–45, 1998.
7. H. L. Bodlaender and T. Kloks. Better algorithms for the pathwidth and treewidth of graphs. Proc. of the 18th International Colloquium on Automata, Languages and Programming (ICALP 91), LNCS 510, pages 544–555, 1991.
8. E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.
9. U. Feige, M. Hajiaghayi, and J. R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM J. Comput.*, 38(2):629–657, 2008.
10. F. V. Fomin, S. Oum, and D. M. Thilikos. Rank-width and tree-width of h-minor-free graphs. *European J. Combin.*, 31(7):1617–1628, 2010.
11. Q.-P. Gu and H. Tamaki. Optimal branch-decomposition of planar graphs in $O(n^3)$ time. *ACM Trans. Algorithms*, 4(3):Article 30, 2008.
12. Q.-P. Gu and H. Tamaki. Constant-factor approximations of branch-decomposition and largest grid minor of planar graphs in $O(n^{1+\epsilon})$ time. Proc. 20th International Symposium on Algorithms and Computation (ISAAC 09), LNCS 5878, pages 984–993, 2009.
13. F. Kammer. Treelike and chordal graphs: Algorithms and generalizations. Ph.D. Thesis, University of Augsburg, 2010.
14. T. Kloks, C.-M. Lee, and J. Liu. New algorithms for k-face cover, k-feedback vertex set, and k-disjoint cycles on plane and planar graphs. Proc. 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2002), LNCS Vol. 2573, Springer, Berlin, pages 282–295, 2002.
15. J. Lagergren and S. Arnborg. Finding minimal forbidden minors using a finite congruence. Proc. of the 18th International Colloquium on Automata, Languages and Programming (ICALP 91), LNCS 510, pages 532–543, 1991.
16. S. Oum. Rank-width is less than or equal to branch-width. *Journal of Graph Theory*, 57(3):239–244, 2008.
17. B. A. Reed. Finding approximate separators and computing tree width quickly. Proc. 24th Annual ACM Symposium on Theory of Computing (STOC 1992), pages 221–228, 1992.
18. N. Robertson and P. D. Seymour. Graph minors. III. Planar tree-width. *J. Combin. Theory Ser. B*, 36(1):49–64, 1984.
19. N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.
20. N. Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Combin. Theory Ser. B*, 52(2):153–190, 1991.
21. N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory Ser. B*, 63(1):65–110, 1995.
22. P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.